Tampere University of Applied Sciences

Design of a system that detects the loss of temperature of a refrigerator using RuuviTag sensor

Jaime Pajares Alonso

BACHELOR'S THESIS

May 2019

Degree Programme in ICT Engineering

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Degree Programme in ICT Engineering

Author: Jaime Pajares Alonso
Title: Design of a system that detects the loss of temperature of a refrigerator using Ruuvi-Tag sensor

Bachelor's thesis 33 pages,
May 2019

Due to the development of 5G technology, Internet of Things is starting to become a reality.
This Bachelor's thesis is based on the design of a system which detects the loss of temperature in a refrigerator. For obtaining this parameter, a RuuviTag sensor has been used and placed inside the household device. For the interaction between the user and the sensor, the system has an Android application for the smartphone.
For obtaining and ensuring the design's best viability and functionality, several techniques and programs were used. These have been explained and showed during the thesis with practical examples for testing purposes.
The IDE of choice is Android Studio and the application is targeted at the Android 4.0.3 operating system.
Also, different problems and difficulties have emerged in general, regarding with the development of the application, however, most of the obstacles have been satisfactory resolved.

Thanks to the design that I propose, it would be possible to implement Internet of Things not only in refrigerators, but also in all old electronic households' devices with no necessity of buying a new one.
The design of the application user interface was created to a point where all the functionalities are completed and described.
Nevertheless, for the implementation of the designed system and all its functionalities, further researches are needed for a perfect performance of the application.

Keywords: 5G, smartphone, application, internet of things

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Ingeniería en Sistemas de Telecomunicaciones

Autor: Jaime Pajares Alonso
Título: Diseño de un sistema que detecta la pérdida de temperatura en una nevera mediante el uso del sensor RuuviTag

Proyecto de fin de grado 33 paginas,
Mayo 2019

Debido al desarrollo de la tecnología 5G, el Internet de las Cosas esta comenzando a ser una realidad.
Este proyecto de fin de grado recoge el diseño de un sistema para la detección de pérdida de temperatura en una nevera. Para la obtención de este parámetro, se ha utilizado un sensor RuuviTag colocado en el interior del electrodoméstico. Para la lectura del parámetro y la interacción con el usuario, el sistema cuenta con una aplicación Android para el Smartphone.
Para este diseño, distintas implementaciones y programas han sido necesitados a la hora de llevar a cabo la aplicación. Estas han sido explicadas y mostradas con ejemplos prácticos a lo largo del trabajo, para testear cada funcionalidad.
El entorno de trabajo utilizado ha sido Android Studio y la aplicación esta codificada para trabajar en el sistema operativo Android 4.0.3 o posterior.
Diferentes problemas han surgido durante la realización del proyecto, la mayoría resueltas de forma satisfactoria. Los que han sido los más difíciles de subsanar se han centrado generalmente en el desarrollo de la aplicación móvil.

Gracias a este diseño, será posible implementar el Internet de las Cosas no solo en neveras, si no también en cualquier electrodoméstico antiguo sin la necesidad de tener que comprar uno nuevo.
Todas las funcionalidades del diseño de la interfaz de usuario han sido creadas y descritas.
Para la implementación de todas las funcionalidades del sistema diseñado, futuros desarrollos son necesarios para un completo funcionamiento de la aplicación.

Palabras clave: 5G, smartphone, aplicación, internet de las cosas

PREFACE

This thesis has been developed in my stay in Tampere during my Erasmus at Tampere University of Applied Sciences. It represents the final work of my Bachelor's degree in Telecommunication Systems Engineering at the Technical University of Madrid.

The professor Mauri Inha offered me the possibility of conducting this project. Is therefore essential for me to place on record my enormous gratitude to him for having been that patient and generous, always willing to solve all the doubts that have arisen.

I would also like to thank my parents and family. I will be eternally grateful to them just for being the way they are. My gratefulness is also to all the people I met during my student period and my friends because they have always supported me.

Tampere, 30 May 2019

Jaime Pajares Alonso

CONTENTS

LIST OF FIGURES

ABBREVIATIONS AND TERMS

ADT	Android Development Tools
App	Application
BLE	Bluetooth Low Energy
CPU	Central Processing Unit
DFU	Device Firmware Update
GUI	Graphical User Interface
IDE	Integrated Development Environment
iOS	iPhone Operating System
IoT	Internet of Things
LED	Light-Emitting Diode
NFC	Near Field Communication
OS	Operating System
RAM	Random Access Memory
RF	Radio Frequency
XML	Extensible Markup Language

1 INTRODUCTION

The use of the technology is increasing with the years, up to the point of having a technology application in almost every life activity.
An example that clearly helps to explain and understand this important evolution and contribution to society's new lifestyle in the last years, are smartphones. Since its impact appearance, technology has changed the way we live, but more interesting how it has improved and shaped the way we now approach our daily activities and decisions. Nowadays it is difficult to imagine a world without them, this is why we should and must continue to think in new ways that could help us to resolve problems that affect our daily capabilities. Having this in mind, the purpose of most of these applications is giving people some help in day to day activities.

Nevertheless, this is just a small example about what technology is able to give to us. Some time ago, Internet of Things (IoT) started to come to light like something future however, this is now becoming a reality. We understand IoT as a digitalization of the physical world, connect every device, gadget, instrument to the network and to each other. This would cause that all the objects that where connected throughout a close circuit, now they can be connected globally by the same network so they can interact between them. The aim of Internet of Things is the same as the majority of technological advances, making life simpler and more comfortable to people life.

The purpose of this thesis is to design a system to the refrigerator that can detect the loss of temperature using Ruuvi-Tag sensor.
The principal aim is to avoid the user problems with its fridge, with any effort on his part so, the user doesn't need to be worried if he left the fridge opened or it has function problems. For this, the user will need to put inside the refrigerator a sensor which will made periodic temperature lectures.
This will be useful for adapting any fridge to the Internet of Things.
Secondary objectives are also presented for the correct development of the project, for example, have a look to Android phone application development and how to implement the sensor with an Android application.

If the refrigerator presents significant problems with the temperature, the sensor will notify the user throughout a notification in the phone. Finally, the sensor will be also able to show the temperature data in a graph through a smartphone application.

2 COMPONENTS

For the accomplishment of the thesis, two components where suggested, Mov sense and RuuviTag. As my idea was to collect temperature data, after researching about both of the sensors, I finally decided to choose RuuviTag. Movesense device is more focused for positioning purposes.

2.1 Ruuvi-Tag

Ruuvi is a Finnish start-up company created by Lauri Jämsä and Henri Hakunti, whose passion for electronics and new innovations took them to create a RuuviTag device. It took 17 years for Ruuvi's team to release the final and main product, which was successfully crowdfunded in 2016.

RuuviTag is an open source sensor beacon platform that is principally designed for all type of customers. It is useful for developers, students, companies…everyone who needs to fulfil any necessity even at home for personal tasks. [5, 6]

There are two versions of the device. For my project I will use RuuviTag+, which contains various built-in sensors that can collect the temperature, relative air humidity, air pressure and acceleration. The data that most concerns me is the temperature data.

Figure 1. RuuviTag+ sensor (Google picture)

2.2 Connections

RuuviTag comes with several specifications. According to the connections, this device has support for multiple protocols including Bluetooth Low Energy (BLE), an integrated NFC antenna, long range RF antenna and Wirepas Connectivity mesh. [3]

The connexion used in this thesis and for the system design is going to be BLE connexion.

2.2.1 Bluetooth

Bluetooth was developed in 1994 by Eriscsson for the replacement of cables. It is a wireless communication technology that allows devices, for example, smartphones, computers or peripherals to transmit information. It uses the 2.4 GHz frequency band, the same than Wi-Fi routers. Comparing to Wi-Fi, Bluetooth utilizes less power and costs less to put into effect. Thanks to this lower power consumption, it is then more difficult to suffer from interferences with other wireless devices that use the same 2.4 GHz band. [8]

Bluetooth is a short-range wireless communication therefore the range is lower than Wi-Fi. The transmission speed is also slower, however, with the latest improvements the speed has increased. [8]

Despite of all the improvements, we can find other limitations when we talk about this connectivity technology. The first one is that the battery will run out before when using Bluetooth. Also, as I said before, the range is usually short, around 10 meters. In addition, similar as with every remote technology, obstacles like walls can make this range even shorter. [8]

According to security, Bluetooth is considered a reasonably secure wireless technology when is used carefully. It uses an encrypted connection. For a normal usage, it is not risky to use Bluetooth. [8]

2.2.2 Bluetooth Low Energy

BLE was introduced in 2004 when Bluetooth 4.0 was released as an alternative to Bluetooth Classic. Its adoption is growing rapidly due to the fact that IoT is expanding. BLE uses the same frequency band of 2.4 GHz as Bluetooth Classic does. [9]

It is important to notice that Bluetooth Low Energy is different as the classic Bluetooth in different aspects. BLE consumes much less power and transmits the information in shorter distances and with less amount of data. The purpose of this connexion type is to transfer small amounts of data, like, for example, beacons (small Bluetooth radio transmitter) [11]. In consequence of its less power consumption, the battery life when using BLE connexions is longer and it is easier and cheaper to implement in devices. [9, 10]

Apple was the first company who implement BLE technology. For Android, Bluetooth Low Energy support came in summer of 2013 with version 4.3 API Level 18.

Nowadays, BLE is mostly used in Internet-connected gadgets with small batteries, based on healthcare, sports, fitness, positioning, temperature sensors…that communicate with smartphones and tablets. [9, 10]

2.3 Software

First of all, regarding the software, I have had to decide which operating system, iOS or Android, was best for the development of the application. Although I am an iPhone owner, Android possible software is better for testing and coding applications. Also, Android OS is present in more smartphones and it is easier for developers to publish its own apps in the store.

IDE software selected for the success of the application is Android Studio.

2.3.1 Android Studio

Android Studio was announced in May 2013 at Google I/O, a developer congress organized every year by Google. It was officially released in December 2014. Android Studio replaced Eclipse ADT as the principal IDE for Android application development. [12]

Android Studio is the official IDE for Android app development. It is based on IntelliJ IDEA, which is a Java integrated development environment for software. [13]

Android Studio offers some features that makes it been better IDE than others. It includes a really good a quick growing Gradle Build Tool. It will help you in increasing your productivity.
Applications built in this IDE are use APK format compilation for compilation. This means that, as I said before, it is easier to submit it in the Google Play Store.
The user interface is very intuitive and easy to use. It has a graphical user interface (GUI) for the design of the different layouts, dragging and dropping the components. This feature is not so important for expert coders, however, that is not my case and it helped me a lot when having an idea when thinking and creating the design.
In terms of system stability, I found Android Studio perfect for the conditions in which I was going to work. I don’t possess a powerful laptop with high amount of RAM and high-speed CPU but, with Android Studio I don’t need it at all.
Finally, Android Studio has an emulator that simulates an Android phone for testing the project. It is also able to connect a physical Android smartphone. [12, 14]

3 IDEA

The main idea is simple, the user must notice if he or she left the fridge opened or it presents a failure when cooling by placing a sensor inside it. The user will be warned about this by an Android app that will notify you when the temperature is above a certain temperature. This app will also provide the user a graph showing the different temperatures depending on the time.

A visual explanation of the idea is shown below:

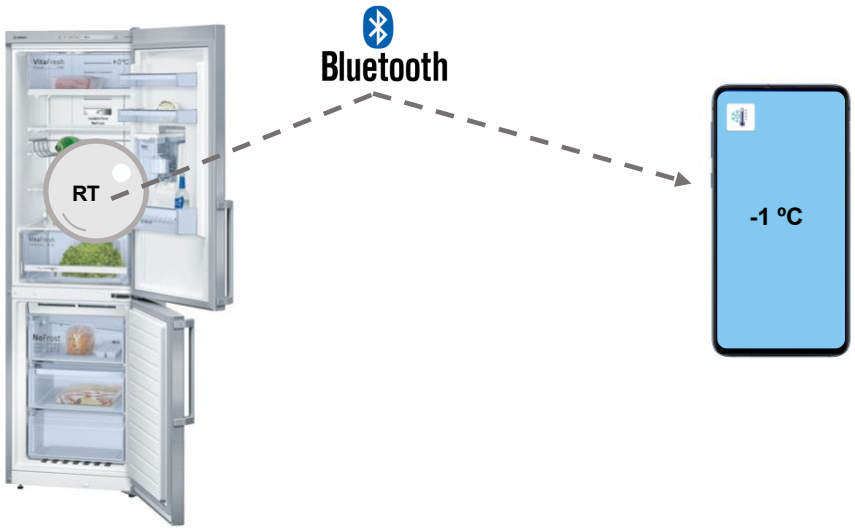


Figure 2. Visual representation of the idea

Figure 2 shows clearly how the idea would work. The RuuviTag+ sensor is placed inside the refrigerator and it would connect with the smartphone via Bluetooth Low Energy. This sensor collects data and sends it to the application, which shows the temperature in real time. The application needs to update the temperature data periodically and show a graph of the temperature in each time is has been updated.

4 IMPLEMENTATION

For the implementation of this idea some steps were necessary. In this point I will explain how to develop the aim of the thesis in the clearest way I can. The implementation will be divided in different points.
On one hand, we have RuuviTag+ sensor, who needs to be updated for a correctly Bluetooth connexion.
On the other hand, we find the hardest part of the project, the application. This section will be the most extensive one, explaining each step needed for obtaining the desired results.

4.1 Firmware update

For a successful connection between the RuuviTag+ and the smartphone via BLE, it is necessary that your device is updated to the last firmware. What is firmware? It is the program that makes the device works as a remote sensor. Thanks to the firmware, the RuuviTag collects data from temperature, relative air humidity, air pressure and acceleration. This firmware has two modes. On one hand, it has a RAW mode which uses BLE to transmit the data. On the contrary, URL mode sends the information to the browser and you can see it in a webpage. [16]. The way selected for my idea is the RAW one, due to the connexion type used.

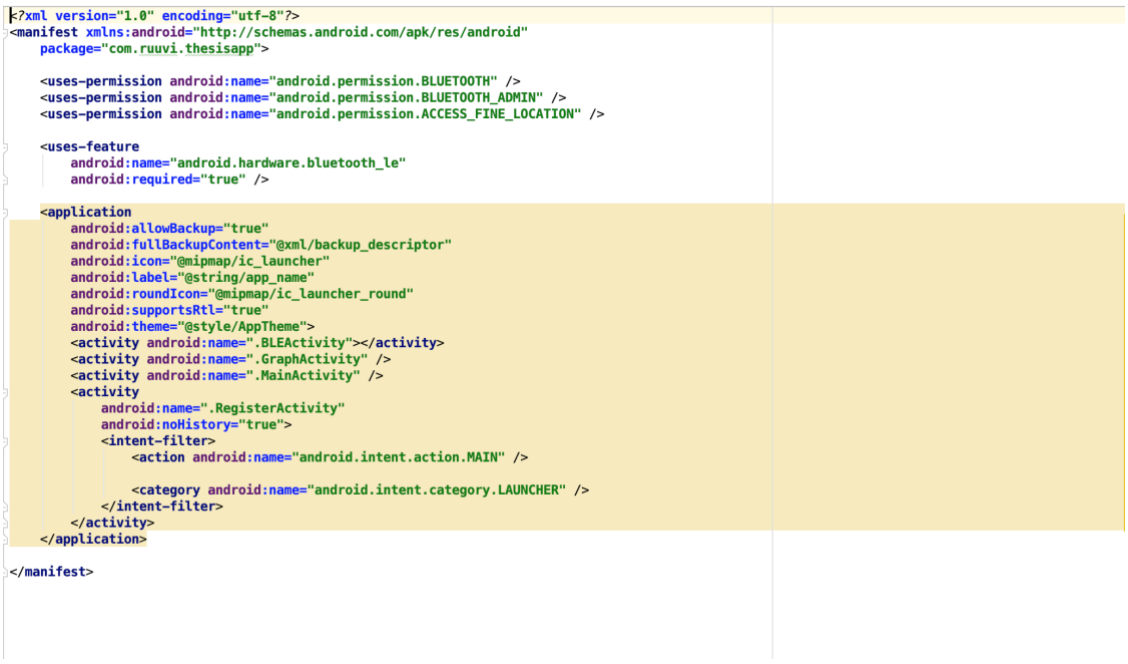
The easiest way firmware can be updated is wirelessly, using NRF Connect application, which you can find in the Google Play Store for Android smartphones. After starting the app, Ruuvi Firmware distribution package needs to be downloaded from the phone and saved in it. This .zip file is found in Ruuvi's official webpage. [15]
Once downloaded, now it's time to start the application mentioned above and connect RuuviTag sensor keeping pressed buttons, "R" and "B". If the red LED stays lit, the device is ready for being searched via BLE. Once connected, if an update is needed, the application will show in the upper right corner a DFU icon. Select the distribution package file type where you saved it, wait for some minutes, and the RuuviTag will be updated. [15]

Furthermore, if no DFU icon is shown, it means that the gadget has the latest firmware installed.

4.2 Developing Android application

Android Studio ADT makes very easy to create new projects. When a project is created, some different files are automatically generated, like activities.java or layouts files. Folders which contains elements of application like images, text and strings are also generated.

In addition, there is a mandatory file for all Android applications. Android Manifest is an XML file which contains all the information about the app that the system needs before the code can be executed. This information are application permissions (contacts, microphone, camera, Bluetooth…), that are shown to the user in Google Play Store before downloading.
Moreover, Android Manifest declares the minimum API version application requires to run, the libraries and the different components like the activities. [17]



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

    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />

    <uses-feature
        android:name="android.hardware.bluetooth_le"
        android:required="true" />

    <application
        android:allowBackup="true"
        android:fullBackupContent="@xml/backup_descriptor"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".BLEActivity"></activity>
        <activity android:name=".GraphActivity" />
        <activity android:name=".MainActivity" />
        <activity
            android:name=".RegisterActivity"
            android:noHistory="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

Figure 3. Android Manifest of my design application

In figure 3 we can observe some of the things mentioned before. For the application I want to design, it is needed to give specific permissions for the

Bluetooth, BLE and access to the location. These permissions were only necessary if it is wanted that my app connects via BLE with other devices.
Activities are also shown, differentiating four, which I will explain during the thesis.

4.2.1 Activities

Activities are the most basic and most used when developing applications for Android. Each screen in an app is an activity. They are formed by two divisions, the logic and the visual part.
According to the logic one, it is a .java file used for coding the activity. Activities contains sections, which are used for going through a number of states. Android Studio creates automatically the section onCreate(). This is compulsory to implement due to it fires when the system creates the activity. There are some other not needed sections, each with a different purpose.
According to the graphic interface, we have layout XML files. In layouts we can find Views components, which provide us of lots of different basic controls. [18, 19]

For my design, the main idea was to divide the application into:

- First, when the app is started, a register screen is shown. This activity is called “RegisterActivity” and it contains a field for writing the name of the user. Once registered, the next activity will be shown. It should not leave the user change the screen if no name is added. The user registered has to be saved in a database for using the app more times with same account.
- Secondly, it is necessary to search for the RuuviTag sensor and connect to it. Activity “BLEActivity” is in charge of doing this.
- Thirdly, once the sensor is connected to the phone the activity “MainActivity” is presented. In this screen, the basic information about the temperature of the fridge is displayed in real time.
- Finally, as the design of the application was that it shows a graph of the temperature, “GraphActivity” is in charge of this. The graph has to show the degrees within the hour data was collected.

4.2.2 User interfaces: Layouts

For the visual user interface, layouts are used. These are also XML files where all the graphic elements that compose the interface are displayed. All these components inherit from the View class. View class is the basic for creating all the components used in the interface. It draws the components in the screen and the user can interact with them. The class ViewGroup inherits from View and is used for designing a structure for more than one View.
As I said above, user interfaces are XML files. This is an advantage when making changes or adaptations of the interface without having to modify the code. It is also much easier and clearer to understand the layout structure looking at an XML file. [20]

For the implementation of my design, one layout for each activity is needed.

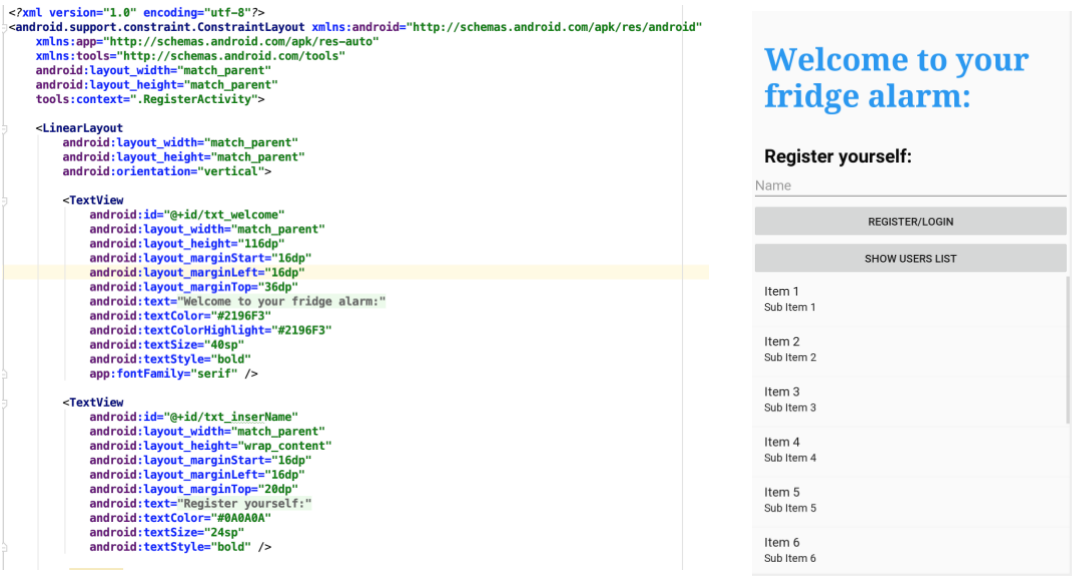


Figure 4. Layout design of “RegisterActivity”

The figure above shows a part of the XML file and the design of the home screen where the user registers itself. It implements two Text Views for the welcome message and the register message, one Edit Text for filling it with the name of the user and one button that will register the user and change to the next screen.

4.2.3 Database

A data base is a set of data which belongs to a same context and is used to store it for a future use.

In Android, the most powerful and used tool for storing data is SQLite. It is very popular due to its small size, no server needed, simple configuration and the most important, for being a free code data base. The typical way of creating, updating and connecting with an SQLite data base is the helper class SQLiteOpenHelper. [21]

For the implementation of the system a database is needed. It is required for two reasons.

I. Store the register users and login them.

II. Store data of the sensor like temperature, time, humidity, air pressure and acceleration. In the case of this design, only time and temperature are required and will be used.

```
public class AdminSQLiteOpenHelper extends SQLiteOpenHelper {
    public AdminSQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version){
        super(context, name, factory, version);

    }

    public void onCreate(SQLiteDatabase DataBase){
        DataBase.execSQL("create table users (name text primary key)");
        DataBase.execSQL("insert into users values ('admin')");
    }

    @Override
    public void onUpgrade(SQLiteDatabase DataBase, int oldVersion, int newVersion) {
        DataBase.execSQL("create table users (name text primary key)");
        DataBase.execSQL("insert into users values ('admin')");
    }

    public ArrayList full_list(){
        ArrayList<String> list = new ArrayList<>();
        SQLiteDatabase db = this.getWritableDatabase();
        Cursor cursor = db.rawQuery( sql: "SELECT * FROM users", selectionArgs: null);
        if(cursor.moveToFirst()){
            do{
                list.add(cursor.getString( columnIndex: 0));

            }while(cursor.moveToNext() && !cursor.equals(db));
        }
        return list;
    }


}
```

Figure 5. Database code

Figure 5 is an example of a database code. In this case, this database is the one used for storing the users when they are registered. It also has a method for saving the users in a list of users.

As we can see it consist of two abstract methods (onCreate() and onUpgrade()) where we put the code necessary for the data base to work properly.

4.2.4 Bluetooth connection

This has been the hardest part of my design for implementing in an Android application due to its complexity.

The development of the Bluetooth Low Energy connection was divided into three phases.

I. Initialization. So as to perform any Bluetooth connexion need to put permissions to the application. These are declared in the Android Manifest file as we can see in figure 3. These permissions are the following:

```
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Figure 6. Permissions for Bluetooth and location

In the next step, the application has to check if the device supports or not BLE technology. It is indispensable that it does because RuuviTag just work with it. If it supported, the application will check is Bluetooth is enabled in the phone and location permitted. If Bluetooth is not enabled, the app should turn it on. If location is not permitted by the app, it will show you a message asking for you to enable the permission.

II. Second phase is to find the BLE devices with Bluetooth adapters using LeScan method. This method scans and returns the devices that are found.

III. The last phase is to connect to the device required, in this case to our sensor. This will enable to collect data from it.

4.2.5 Notifications

An important part in the design of the system is that the user has to receive a notification when the fridge is not working properly. This has to be a push notification, displayed in the notifications bar of the phone.

Notifications provide some information in a short period of time about any event in the application while it is not in use.

The class that is in charge of managing the notifications in the notification bar is NotificationManager(). [22]

```
//Create a notification
public void sendNotification(){

    //Set the notification content
    NotificationCompat.Builder builder = new NotificationCompat.Builder( context: this)
            .setSmallIcon(R.drawable.notification_icon)
            .setContentTitle("Alert")
            .setContentText("The temperature is too high")
            .setPriority(Notification.PRIORITY_HIGH)
            .setDefaults(Notification.DEFAULT_ALL)
            .setAutoCancel(true);

    NotificationManager mNotificationManager = (NotificationManager) getSystemService(this.NOTIFICATION_SERVICE);


    mNotificationManager.notify( id: 1234, builder.build());

}
```

Figure 7. Notification code

Figure 7 shows the notification code of the implementation of my design. When the application needs to alert the user, the code above produces the notification and it is displayed in the screen throughout the notification bar.

5 Application

Following I am going to explain the different parts of the design of the Android application for implementing the system that detects loss of temperature in a refrigerator using RuuviTag sensor.

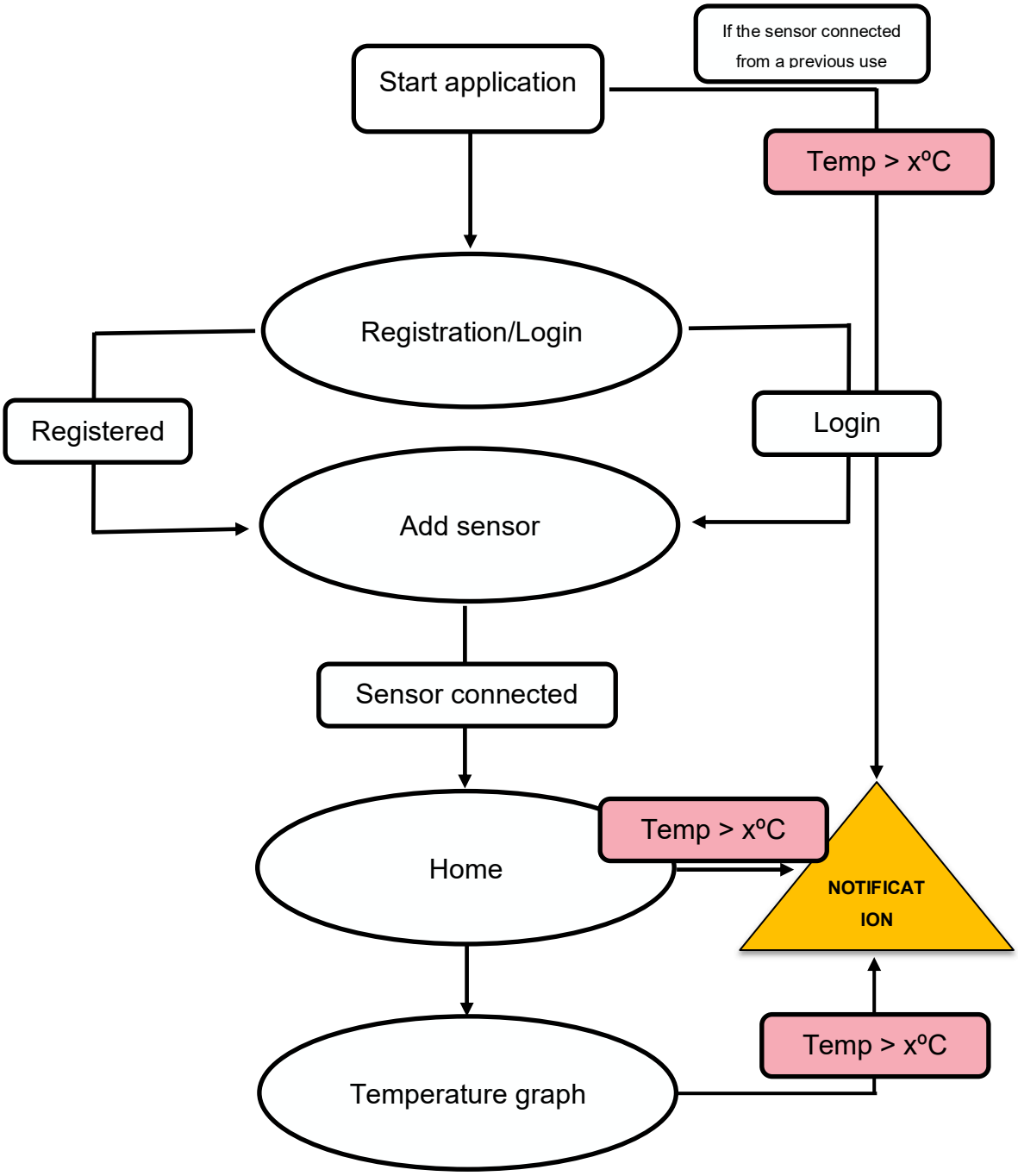


Figure 8. Diagram of the application design

5.1 First screen

When starting the app, a welcome message should be shown. Down this message, another one asking for the user to register has to be displayed, with a place for writing the name. Finally, two buttons (register/login and show users list) are also displayed. The different actions of the register screen should be the followings:

- If the user has never register before, he or she has to write its name and he will be instantly registered by clicking the “register/login” button.

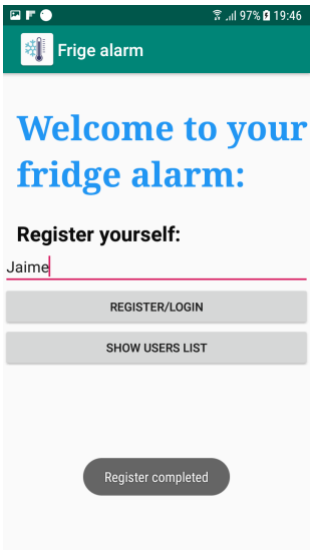


Figure 9. Register screenshot

- If the user had already registered, he will login in the app writing its name and pressing “register/login” button.

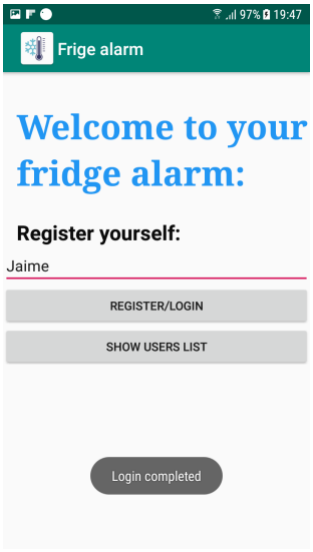


Figure 10. Login screenshot

- If the user doesn't remember if he had registered or with which name, by tapping the "show users list" the application will display all the previous registered users. If its name is in the list, the user can press it and login into the app.

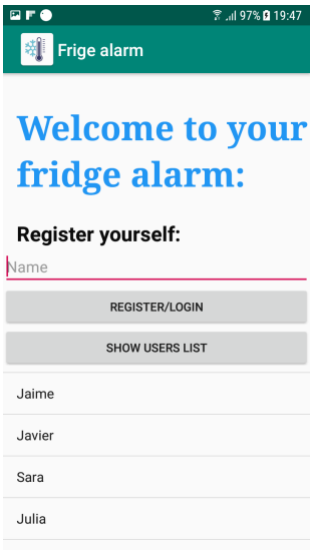


Figure 11. User List screenshot

In the cases where the user wants to register or login by writing the name, no empty blank should be leaved. If the user wants to register with no name, the app has to warn the user with a message.

5.2 Second screen

Once the user has been registered or logged in, the app has to send him to the second screen. Now it is time to search for the sensor and connect it via Bluetooth. This part of the app should do the following:

- Firstly, if the Bluetooth is not enabled in the phone the user would see a notification asking for permission for turning on the Bluetooth.

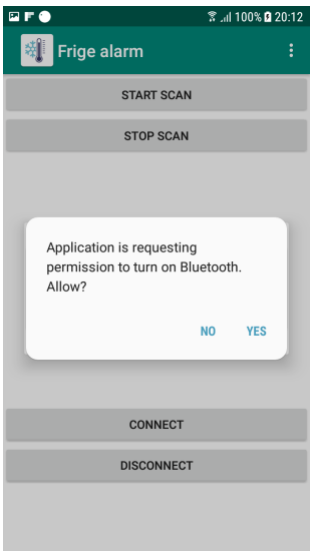


Figure 12. Bluetooth permission screenshot

- Secondly, the app also needs location access. If location is not permitted in the phone, another notification should appear. However, this time is the user who needs to go to the permissions of the apps settings and enable it.

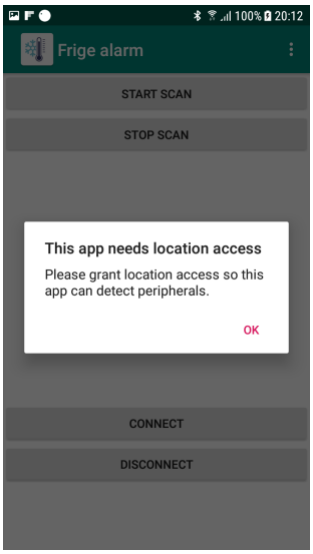


Figure 14. Location permission screenshot

- Thirdly, once Bluetooth and location permissions are enabled, a screen with four different buttons (Start scan, Stop scan, Connect and Disconnect) are displayed. By tapping the “Start scan” button, the app has to search for BLE devices around the phone.

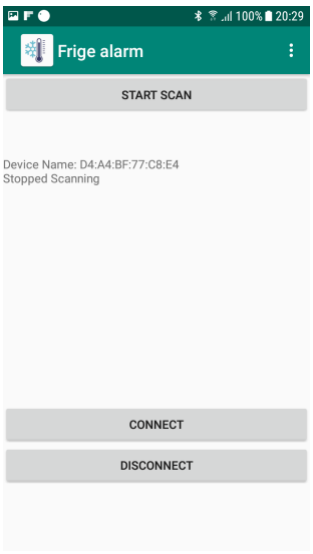


Figure 15. Start scan screenshot

- Connecting to the device will produce a change of screen. If the device is successfully connected, a notification is displayed as we can see in the next figure:

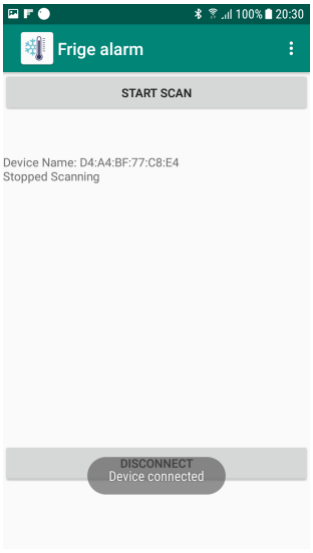


Figure 16. Device connected screenshot

- Similar to the previous point, if the device disconnects from the phone a notification is also displayed.

5.3 Third screen

The third screen is the main one, where the temperature value that the sensor reads has to be shown.

- The user can see it automatically when the device is successfully connected. The value should be updated.

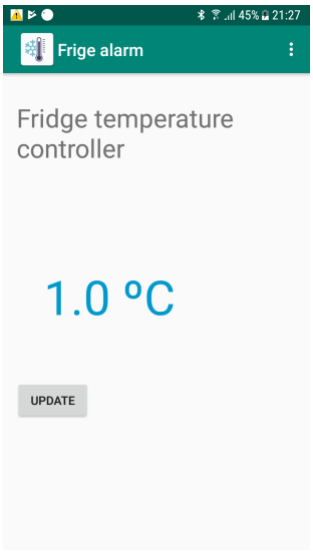


Figure 15. Home screen

- If the temperature read by the RuuviTag is too high for a refrigerator (above 5 ºC), a warning notification alerts the user displayed in the notification bar.

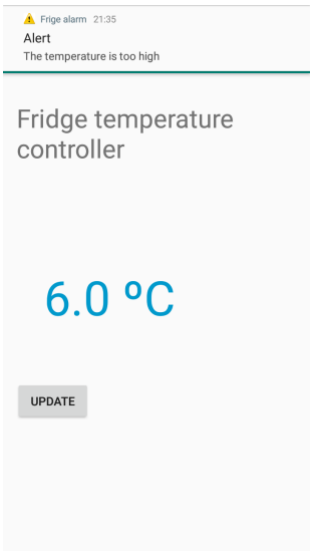


Figure 16. Home screen with notification

- In the upper-right corner, a menu can be displayed by the user by tapping the three points. This menu has to be able to change between the add sensor screen (second one), the graph screen (fourth one) or staying in the same. This menu appears also in the second and fourth screens.

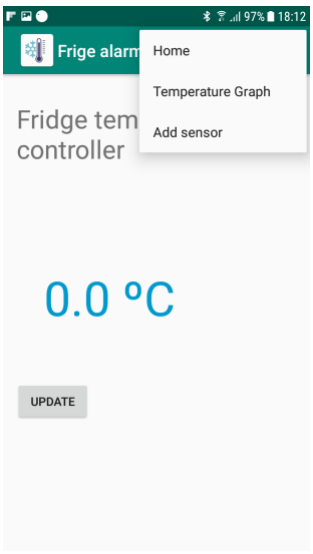


Figure 17. Menu displayed in Home screen.

5.4 Fourth screen

The last screen designed for the system is the temperature graph depending on the time. It has to be able to read data form the sensor, draw it in the graph and store it for making a complete one.

- The graph design is a linear graph. The temperature is in the Y axis and the hour data has been read from the sensor in the X axis.

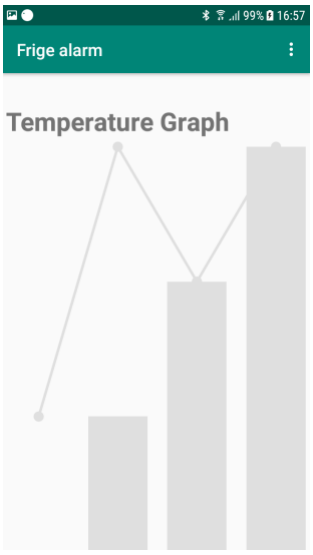


Figure 18. Temperature graph screen

- The graph is overwritten every day, so a different graph is made every day.

6 CONCLUSIONS AND FURTHER RESEARCHES

6.1 Conclusions

The main objective of the thesis was to design a system which is able to detect a loss of optimal working temperature in a refrigerator using RuuviTag sensor. A series of secondary aims were also considered, such as implementing IoT technology to old daily tools (in this case a fridge), familiarizing with Android development and use of external sensor for designing an Android application. These goals were met. Both main and secondary objectives of the project are achieved and explained during the project, presenting the different implementations and technologies needed for them.

In addition to the main purpose of the thesis, and implementation of the phone application has been done. However, this has not been completed due to the requirement of high programming skills and understanding for creating a completed and functional application.

The most challenging part of the application was the Bluetooth implementation and data collection of the sensor. According to this, half of the work has been done. The app is capable of searching and connecting to Bluetooth Low Energy sensor.

Because this last part is not completed, I have not been able to create the whole application. Third and fourth screens presented are examples of the implementation needed, but not as they have to work in my design.

6.2 Further researches

Further research and development that are of interest after the fulfilment of this thesis are explained along this section. Therefore, the completion of this project can serve as a starting point for further implementations of IoT to other types of devices.

For this project, the final product and design is focus in the Android operating system, nevertheless a further development could be doing it also for iOS.

Regarding to the Android application, the principal further development should be finishing it. In addition, lots of additional implementations and improvements can be introduced in the design of each screen. Even more screens can be created with different intentions.

Additional features and improvements that could be implemented in future developments are:

- Since the sensor collects different data, not only temperature, home screen can also show the different parameters the user demands.
- A screen that gives to the user the possibility to adjust the warning for the different parameters the sensor collects.
- General design of the application can be improved and make it more attractive visually.
- It can be a good idea to show more graphs than just the one of the temperatures.

REFERENCES

[1] A fondo: ¿Qué es IoT (El Internet de las cosas)?, DomoDesk, 3.02.1999, [retrieved: 17.03.2019] Available at: https://www.domodesk.com/221-a-fondo-que-es-iot-elinternet-de-las-cosas.html

[2] Aplicaciones móviles: Que son y cómo funcionan, Comisión federal de comercio, September 2011 [retrieved: 17.03.2019] Available at: https://www.consumidor.ftc.gov/articulos/s0018-aplicaciones-moviles-que-son-y-como-funcionan

[3] Lauri Jämsä, RuuviTag Datasheet 4/2018, 8.02.2017 [retrieved: 6.05.2019] Available: https://blog.ruuvi.com/datasheet-52fb00265c60

[4] Luis del Valle Hernández ,99. Arquitectura IoT, prototipando los dispositivos del futuro, programarfacil.com, [retrieved: 17.03.2019] Available: https://programarfacil.com/podcast/arduino-wifi-proyectos-iot/

[5] Ruuvi, Hello friend, we're Ruuvi [retrieved: 25.02.2019] Available: https://ruuvi.com/about-us/

[6] Ruuvi, What is RuuviTag? [retrieved: 25.02.2019] Available: https://ruuvi.com/manuals/tag/what-is-ruuvitag/

[7] WWWhatsnew, El Internet de las cosas | ¿Qué es y cómo funciona?, 5.01.2016, [retrieved: 17.03.2019] Available: https://www.youtube.com/watch?v=uY-6PcO96Bw

[8] Melanie Pinola, Bluetooth Basics, 04.02.2019 [retrieved: 7.05.2019] Available: https://www.lifewire.com/what-is-bluetooth-2377412

[9] ShopperTrak, What is BLE (Bluetooth Low Energy)?, 02.16.2015 [retrieved: 7.05.2019] Available: https://www.shoppertrak.com/article/what-is-bluetooth-low-energy-ble-bluetooth-smart/

[10] Margaret Rouse, Bluetooth Low Energy (Bluetooth LE), 11.2014 [retrieved: 7.05.2019] Available: https://internetofthingsagenda.techtarget.com/definition/Bluetooth-Low-Energy-Bluetooth-LE

[11] Kontakt.io, What is a beacon? [retrieved: 7.05.2019] Available: https://kontakt.io/beacon-basics/what-is-a-beacon/

[12] Margaret Rouse, Android Studio, 10.2018 [retrieved: 7.05.2019] Available: https://searchmobilecomputing.techtarget.com/definition/Android-Studio

[13] Developers, Meet Android Studio [retrieved: 8.05.2019] Available: https://developer.android.com/studio/intro/

[14] Mejul Rajput, Why Android Studio Is Better For Android Developers Instead Of Eclipse, 20.05.2015 [retrieved: 8.05.2019] Available: https://dzone.com/articles/why-android-studio-better

[15] Ruuvi, Firmware Update, [retrieved: 9.05.2019] Available: https://lab.ruuvi.com/dfu

[16] Ruuvi, Official RuuviTag Firmware, [retrieved: 9.05.2019] Available: https://lab.ruuvi.com/ruuvitag-fw/

[17] Developers, App Manifest Overview [retrieved: 9.05.2019] Available: https://developer.android.com/guide/topics/manifest/manifest-intro.html

[18] Developers, Introduction to activities [retrieved: 9.05.2019] Available: https://developer.android.com/guide/components/activities/intro-activities

[19] Academia Android, componentes de una aplicación, 19.03.2016 [retrieved: 9.05.2019] Available: https://academiaandroid.com/componentes-aplicacion-android/

[20] Academia Android, Interfaces de usuario: layouts, 19.05.2016 [retrieved: 9.05.2019] Available: http://academiaandroid.com/interfaces-de-usuario-layouts/

[21] La Geekipedia de Ernesto, Database SQLite in Android, 27.03.2018 [retrieved: 9.05.2019] Available: https://www.youtube.com/watch?v=TxkdWX3UaNk&t=59s

[22] Developers, Create a notification [retrieved: 21.05.2019] Available: https://developer.android.com/training/notify-user/build-notification