FACULTY OF SCIENCE, ENGINEERING AND COMPUTING

School of *Computing and Information Systems*

BSc DEGREE
IN
Computer Science

PROJECT DISSERTATION

Name: Juan José (Pepe) Torrejón Rodríguez

ID Number: K1445897

Project Title: Health Wristband with Arduino

Project Type: Build

Date: 09/04/2015

Supervisor: Dr. Robert Mellor

Kingston University London

Did you submit a draft of your dissertation to your supervisor?	Yes
Did you receive feedback from your supervisor on any submitted draft?	Yes

Plagiarism Declaration

The following declaration should be signed and dated and inserted directly after the title page of your report:

Declaration

I have read and understood the University regulations on plagiarism and I understand the meaning of the word *plagiarism*. I declare that this report is entirely my own work. Any other sources are duly acknowledged and referenced according to the requirements of the School of Computing and Information Science. All verbatim citations are indicated by double quotation marks ("…"). Neither in part nor in its entirety I have made use of another student's work and pretended that it is my own. I have not asked anybody to contribute to this project in the form of code, text or drawings. I did not allow and will not allow anyone to copy my work with the intention of presenting it as his or her own work.

Date **08/04/2015**

Signature 

Index

1. Introduction

This wristband is oriented to elderly, overweight people with high cholesterol, and generally unhealthy people.

On one hand, we can easily search on Internet for the availability equipment to check vital signs, but these are too expensive and many people haven't got enough money to buy them, or even part of this equipment. The point objective is to use some of the cheapest devices and sensors in the world to make one "big" device which can measure vital signs and alert people to any dangers.

It's important to make these checks constantly because many people can have problems or die very quickly if something within their body fails and they aren't aware. For example, if someone lives alone, and has a heart attack, this wristband will be able to warn a nearby hospital or a contact, depending on the configuration of the device.

It is easy to find news articles online about the dangers of an unhealthy population. The "World Health Organization" says "**Overweight and obesity are leading risks for global deaths. Around 3.4 million adults die each year as a result of being overweight or obese. In addition, 44% of the diabetes burden, 23% of the ischaemic heart disease burden and between 7% and 41% of certain cancer burdens are attributable to overweight and obesity.**"[1]

On the other hand, the equipment available is all stationary and must be kept in a fixed place. These devices are not wireless and cannot be used remotely.

As well as that, this equipment is complex and hard to use, meaning a trained professional is required to run the device and interpret the data, causing staff shortages and people waiting a long time to be seen.

In short, the principal objective of this project is to make a cheap and easy-to-use device to control the vital signs remotely, and alert whoever people want, for example another person or any hospital near.

The wristband can be created using a 3D printer meaning it's cheaper, easy to modify and alter so that it can meet anyone's personal requirement. The 3D printer can also be used to create the support for the boards.

Looking again on Internet to look for similar projects or products, there's one product similar to this, but it hasn't the feature to communicate with any Android device, only with hospitals. Also, there are some expensive wristbands which can communicate with iPhone via Bluetooth only, still doesn't appear any product which can communicate with Android devices or even via GPRS. Apart from that, devices which exist now with this purposes, are all of them oriented to people who do some sports.

[1] Article from (World Health Organization, n.d.)

One of these devices is a wristband made by the Foxconn company, but only for IPhone and for people who do sports, because only can control the pulse rate and the respiratory rate, without any communication with other person or hospital.

To sum up, we can find devices like the proposed in this project, but with different purposes, or with only one of the objectives achieved. **This introduction was taken from the proposal, because nothing has changed and the project is still the same, with the same objectives and oriented for the same people.**

1.1. System Overview

The system will consist in three totally different parts. These parts are hardware, software and 3D model.

Hardware will consist on the different boards and the sensor the wristband will have. This part will have an Arduino board, pulse rate sensor (made specifically for these boards), Bluetooth module (in order to communicate with smartphones) and some LEDs that will indicate the wristband status. Apart from that, the wristband will be power supplied with some button batteries.

Software part will be an Android app for smartphones, made as well in this project in order to cover different technologies in this Final Year Project. This app will be compatible with any Android device which has Gingerbread system (2.3 Android version) or above.

Finally, the 3D model will be a wristband made on Google Sketchup and printed with any 3D printer. At the beginning, the wristband will be printed with flexible material if the material is available.

1.2. Objectives

1. One of the aims of this project is to make the cheapest wristband to release out. The components will be used in this project are Open Hardware, so this means everyone is capable to do these components on their own, without any copyright or something similar. In this case, Chinese components will be used because of their price, the components are cheaper than the “originals”, although the quality is not the same as the “originals”.
2. Make and Android app which reads the data sent by the pulse rate sensor, in this case the user’s BPM, and notice somebody’s phone the user has introduced in the app.
3. Make the wristband design in 3D before the VIVA if it is possible. This is the most difficult part of this project, because 3D printing is really new and there are less information about it and its designs.
4. If there is time after making this report and before submitting it, connect the app with a database in order to save users data.
5. This project will be one part of another bigger project which it will be made after this one and try to release this product out.

1.3. Report Organisation

Part 1: Introduction	Just an introduction to the project, aims & objectives and a brief overview about the future project.
Part 2: Literature Review	Information about the different technologies will be used on this project, their histories, advantages, disadvantages and comparison with other similar technologies.
Part 3: Analysis & Methodology	The final Gantt chart followed in this project, time management and use case diagram will be shown
Part 4: Design	Design of both hardware and software things. Schematics for the hardware part, and a diagram about what things will be needed for the app will be shown. 3D model design of the wristband.
Part 5: Implementation	Photos of the wristband prototypes and final version. App screenshots and most relevant pieces of code. Photos about printing on the 3D printer.
Part 6: Testing & Evaluation	Testing both parts, communication between the wristband and the computer, the wristband and the mobile phone running the app.
Part 7: Critical Review & Conclusion	Talking about the project in general, problems encountered and their solutions.
Part 8: Bibliography	References of websites and books used for this project.

2. Literature Review

In this section the information about the technologies used in this project will be analysed. As far as we know, Arduino, Android and 3D Printers. Some books and videos have been used for researching information, but the most used resource in order to find all these information was Internet. Everything will be referenced and listed in the bibliography.

2.1. Arduino

2.1.1. History

Arduino started as a project for students in 2005. It began in the Interaction Design Institute in Ivrea, Italy. Massimo Banzi was the leader of this group, and made this possible on that time. It started as a project for those students who were studying at the Institute, in order to make a cheaper board and a path to build different things. They used to use the "Basic Stamp" board, which cost around 70 pounds each one, an extremely price for students. So one of the main purposes of Arduino was build a new board to replace the "Basic Stamp" board. As every project at the beginning, this hadn't support of any shop, manufacturer, etc. (Valera, 2010)

Nowadays many Arduino boards have been released, so it is possible to make any project, imagination is our limit. Apart from that, there are other manufacturers who made another kind of boards with same purposes as Arduino; the clearest example is Raspberry Pi.

2.1.2. Explanation of this technology

Arduino is an open-hardware based platform. What open hardware is? We can think about open-source software, open-source software is computer software with its source code is available free and can be used and modified by anyone. This type of software is always free. So, open hardware is the same as open-source software, board schematics are available free on Internet, and any company or person can produce the same boards, much cheaper than the original one. Their purposes are to take electronics to every classroom in this world, and try to be the easiest way for everyone to make projects than this.

Secondly, there are multiple boards from Arduino Company to choose. These boards cost almost 20€ (£15 approximately), but a Chinese retailer was chosen to buy the boards and components used on this project in order to achieve the best price for the wristband.

Apart from that, when it is necessary to choose any Arduino board for using on any of these type of projects, it is needed to consider every aspect of the project. This means what devices are going to be used, how many of them, and the general purpose of the project, because there are plenty of Arduino boards released already, changing the specifications on every board.

2.1.3. Arduino vs. Raspberry Pi

In this section, the advantages and disadvantages of using Arduino board on this project will be explained, and comparing it with Raspberry Pi.

2.1.3.1. Advantages of using Arduino

1. Arduino is open-hardware, so it is possible to create your own board using the schematics uploaded on Internet, or modify them in order to create another board with specific requirements of any project.
2. It is really cheap; it is possible to buy an Arduino board for beginners from 18 pounds. Being open-hardware, Chinese manufacturers make these boards much cheaper, so it is possible to find these boards in different websites from the price of 2 pounds.
3. It is easy to implement. Arduino code is in C++ language, one of the most important programming languages. This language is easy to understand and it is possible to start programming having learnt a few C++ lessons.
4. These boards are totally plug & play, so they are compatible with any actual OS, Windows, Linux or Mac OSX. Also, the IDE used for programming these boards is compatible with the same systems.
5. Most of them has USB interface, combined with the above advantage, makes Arduino one of the most OS-compatible board in this world.
6. Any device, (for example engines, LEDs, sensors, and a large etc.) can be plugged in any Arduino board and make it work easily. Is no necessary any power supply for the board, because the same USB connector power supply the board and the components installed on them.

2.1.3.2. Disadvantages of using Arduino

1. Arduino is really limited on its Inputs/Outputs (known as I/O in the rest of the document). It is not possible to be enough for some kind of project that require many devices.
2. Arduino IDE is less optimized, many times the IDE doesn't respond because it doesn't find any Arduino board connected.
3. Serial communication is really strict. You need to specify the serial data transfer for each component which uses this type of communication. In this project, a Bluetooth module will be used, which works from 9600 baud. But if the baud rate is not specified, it is not possible to obtain the results expected.

2.1.3.3. About Raspberry Pi

Raspberry Pi is a single-board **computer** developed here, in the United Kingdom by the Raspberry Pi Foundation. This board is a computer itself, so it has his own processor and RAM memory, apart from USB connectors, RJ-45 connector, SD slot, VGA, HDMI and some audio I/O. Those elements including the GPIO slot compose the Raspberry Pi. The possibilities with this computer are really better than the

possibilities given by the Arduino boards, but it has some disadvantages which makes this board not to be used at least on this project.

One of the main disadvantages for this project using the Raspberry Pi is the energy consumption. It is much more than any Arduino board, and in this case, the wristband will be power supplied with button batteries. These button batteries do not have the enough voltage and amperage to power supply a Raspberry Pi.

It has no internal memory, so it is necessary to buy an external memory, in this case an SD card in order to save any data and also install Raspberry Pi system.

To finish this, Raspberry Pi is much more expensive than Arduino boards, so it is impossible to reach one of the objectives of this project. It costs around 35 pounds.

In addition, apart from Arduino boards are much cheaper than Raspberry Pi, it is true that Raspberry Pi has much more functionalities than Arduino Boards, but for this Project, an Arduino board will be perfect for our purposes. Afterwards, if more functionalities are needed on our Arduino board, it is possible to add whatever it is wanted. These "functionalities" can be added thank to boards called "Shields". Every "Shield" has one function, one purpose, for example, in this case it is going to be used a Bluetooth Shield, this is a module apart from the Arduino main board.

2.2. Android

2.2.1. History and explanation

Android was an operating system for mobile devices totally unknown until Google bought the company who made it in 2005. There were a few rumours until 2007, when 'Open Handset Alliance' was released, which group were formed by most of the mobile phones manufacturers, chipsets and Google itself. First Android version was released on the same year with the SDK, in order to help developers to create their own apps.

Android is the most used operating system in mobile devices, such as phones or tablets. It started in 2011 when Google released the 3.0 version, which was oriented only for tablets. That was the beginning of using Android in bigger devices than mobile phones.

So we can assume that Android is an operating systems only for mobile devices, but not only for mobile phones, we can include tablets, laptops and even radio cars. That is one of the best advantages of Android, it is open source, and can be used by any device which has the minimum requirements to support it.

2.2.2. Android vs. iOS

In this part, the advantages and disadvantages of using Android system for this project will be described and the reasons why it was chosen instead of iOS for Apple devices.

2.2.2.1. Advantages of using Android

1. Android is open source. As mentioned before, Android is based on Linux Kernel, which is open source as well and could be manipulated as the developer wants. Also means that every person in this world can make an app for this system totally free.
2. Create apps for this system is really easy, even if you do not know anything about programming.
3. Android allows to install any app the user wants.
4. Android has the biggest developers' community, so if you need something specific, check "Play Store" before you start programming, it is possible that the app you want to develop is created.
5. Being this operating system open-source based, each developer does not need to pay anything if he wants to make any app. That is one of the reasons which Android is better than iOS in programming.
6. Android uses JAVA for developing apps. This language is one of the most languages used by developers.

2.2.2.2. Disadvantages of using Android

1. Tools for developing apps are really poor. There are a couple of IDEs, but they are not final versions of it, and it is possible to get some crashes while you are programming.
2. Android is multitasking. This is a really good feature of this operating system, but it has other side completely different, battery life of mobile phones comes down if they have many apps opened.
3. Android is totally less intuitive, many people really have problems configuring their terminals and need to check internet every time they have a problem or they want to configure something and they do not know how to do it.
4. Android is totally fragmented. This means that each mobile phone has its last version of Android and the files of the previous versions in order to not lose apps compatibility.

2.2.2.3. Android vs. iOS

We know iOS as the operating system of Apple mobile devices, such as iPhone or iPad. This project is totally oriented to Android devices, because development for this operating system is totally free and its licenses as well. If you want to make apps for iOS, you need to pay for its licenses per year, almost 65 pounds. Apart from that, benefits for developers are less than the benefits for Android developers.

In Android, developer benefits are 70%, and the another 30% for Google. In iOS, 50% are for the developers, 30% for Apple, and 20% taxes which the developers need to pay.

In second hand, Apple devices are really expensive compared to Android devices. It is easy to buy an Android device for 80 pounds. Apple devices cost from 539 pounds.

In summary, if you want to get more benefits on your apps or not to pay anything for publishing them, Android is the best option to start programming apps. But on the other hand, if you want to make benefits as much as possible, is to make your apps for both operating systems, because it is almost impossible to force users to use a platform for our apps.

2.3. 3D Printers

Last technology will be used in this project, as much as possible, are 3D printers. This technology began famous a couple of years ago, but it was in 1983 when Chuck Hull design the first method of 3D printing, the stereolithography (a.k.a SLA).

Nowadays 3D printing is being more and more famous. It is possible to print whatever you want in the way you want, even food. In the future, this technology must be in every home, because gives many chances to fix whatever people want and need, but actually is really expensive to obtain one of these printers.

The only story is possible to be mentioned about this is the stereolithography thing, this technology is progressing actually and there is less information about them on Internet.

2.3.1. Advantages of using 3D Printers

1. The wristband can be personalised as the user wants it, even if the user wants to include or remove anything on it.
2. Can be modified and improved for future revisions of the model easily.
3. Can be printed in different type of materials, such as plastic, flexible plastic, metals, titanium, bronze. (3D Printer Help, 2013)

2.3.2. Disadvantages of using 3D Printers

1. 3D Printers are really expensive, so the manufacturer, who wants to make this, needs to invest money on this technology.
2. Make any object in 3D could be really difficult, a specialist 3D designer will be needed in order to make any good product.
3. Printing the models usually takes so much time, more or less 2 or 3 hours depending the size, thickness, etc.

2.3.3. Conclusion

In conclusion, this will be a good experience for this project, researching about the 3D printing, making the wristband in Google SketchUp, and experience what the professionals do in this field.

2.4. Critical Review

Finding all the information about the history of the technologies will be used on this project it was quite difficult, because there is only technical information about these things, but it is always important the history of them and how it started before they came famous.

This was a quick view about the different technologies will be used on this projects, justifying why every technology is going to be used, giving the advantages and disadvantages, and comparing to other similar technologies.

In summary, writing this Literature Review, many things have been learnt about the history of these technologies, how they started and how they are famous now. Apart from that, researching about these technologies, it is possible to see how many people support these technologies and work with them.

3. Analysis & Methodology

3.1. Analysis

This project has 3 totally different parts. As known, hardware part, software part and 3D design part. The best way to develop this project is to complete each task before starting a new one. Apart from that, it is necessary to have the hardware part before developing and finishing the Android app, because if the app is developed before the hardware, it is impossible to try it completely.

The best way to know how the app will work is making a use-case diagram, it explains what every screen does, what the user will see and expect from it. Also, it will be useful to know what objectives are needed to achieve when the Android app is being developed.

3.1.1. Use-Case Diagram (Functional Requirements)

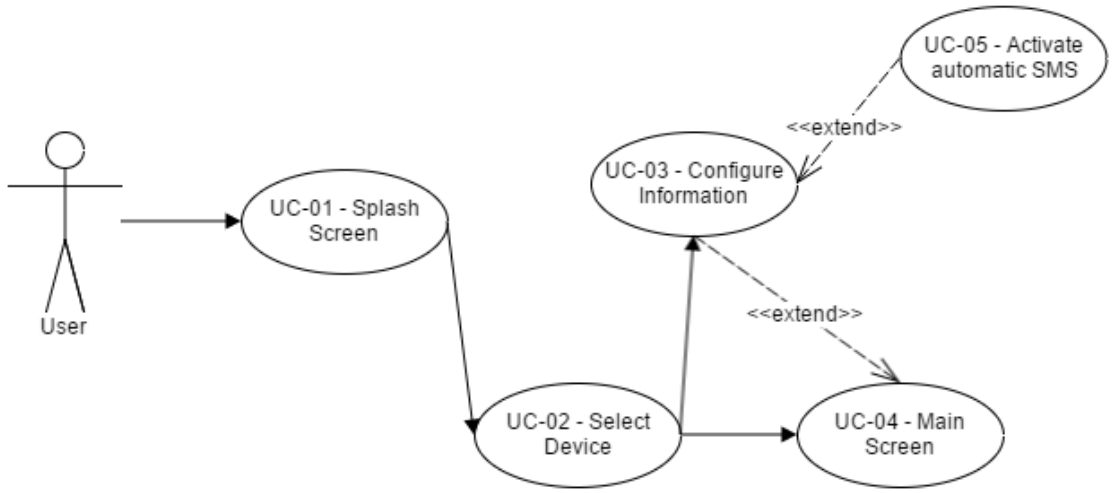


Figure 1: Use-case Diagram

Name	**UC-01** – Splash Screen
Description	Splash screen appears in the mobile phone loading the app
Precondition	Having the app installed in the mobile phone
Post-condition	App loads correctly and shows the device list screen afterwards
Main flow	**Step 1** – User open the app
Alternate flow	There is no alternate flow

Name	**UC-02** – Select Device
Description	The user select the Bluetooth device from the device list
Precondition	App loaded correctly
Post-condition	Connection between the wristband and the mobile phone
Main flow	**Step 1** – User selects the wristband on the list
Alternate flow	There is no alternate flow

Name	**UC-03** – Configure Information
Description	User saves the username, blood type and number phone
Precondition	Wristband connected to the mobile phone
Post-condition	Information saved
Main flow	**Step 1** – User type the information on each field for it **Step 2** – User click on "Add Information Button"
Alternate flow	There is no alternate flow

Name	**UC-04** – Main screen
Description	The mobile phone shows user's BPM
Precondition	Wristband connected and information saved
Post-condition	There is no post-condition
Main flow	**Step 1** – The mobile phone automatically shows user's BPM
Alternate flow	There is no alternate flow

Name	**UC-05** – Activate SMS
Description	Activate automatic SMS
Precondition	User's information saved
Post-condition	Automatic SMS activated
Main flow	**Step 1** – User needs to go to the settings screen **Step 2** – User click on the Activate SMS checkbox **Step 3** – User click on the mobile's phone "Back" button
Alternate flow	There is no alternate flow

3.1.2. Non-functional requirements

The app must achieve the following non-functional requirements:

Name	NFR-01 – Work properly
Description	The app must work properly in each Android mobile phone with minimum 2.2 Android version installed.

Name	NFR-02 – User interface
Description	The app must have a clear, comprehensible and easy to use interface. The user doesn't need to know much about mobile phones and their apps to use this one.

Name	NFR-03 - Performance
Description	The app should work fast and efficiently.

3.2. Methodology

3.2.1. Development methodology

The lifecycle model used in this project is the waterfall lifecycle model. This is the most common lifecycle used for developing any project. In this case, it is used this lifecycle model because it has 3 different parts, and as it is possible that the hardware part is necessary for trying the software part.

Some advantages of this lifecycle model are:

- It is simple and easy to use, as mentioned before, this is the most common lifecycle used in software development, or any project.
- Every phase specified for the project is completed once it is finished.
- It is the best choice for small project, were the requirements are totally understood.

On the other hand, some disadvantages are:

- As mentioned before, it is not possible to have a preview of each phase, every phase can be viewed once it is finished.
- It is possible that the final project doesn't have all the requirements specified for the project.

Apart from that, the software development has been combined with a repository on Internet, in order to have a version control of it. In this case BitBucket was the repository used for this project. Also, it was impossible to have a repository for the 3D wristband design, so in this case, Dropbox was used to have a version control of it.

3.2.2. Gantt Diagram

In the project proposal, the Gantt chart showed was the image below:

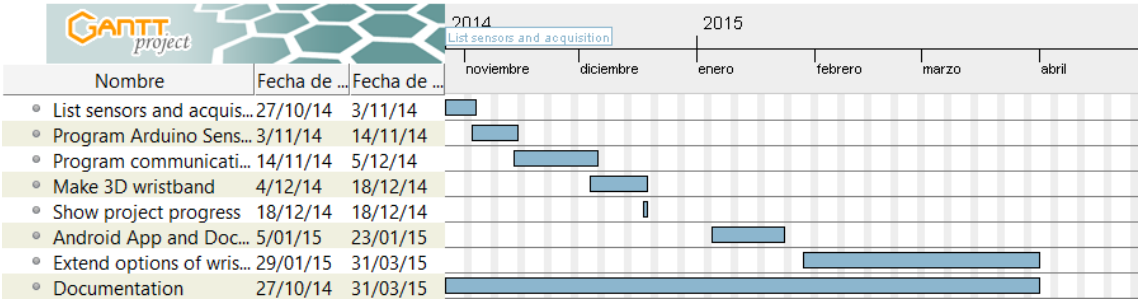


Figure 2: Proposal Gantt Chart

Finally, dates has been changed because different phases of the project has been made before or after than expected. As we can see in the final Gantt chart, the different dates used for the project:

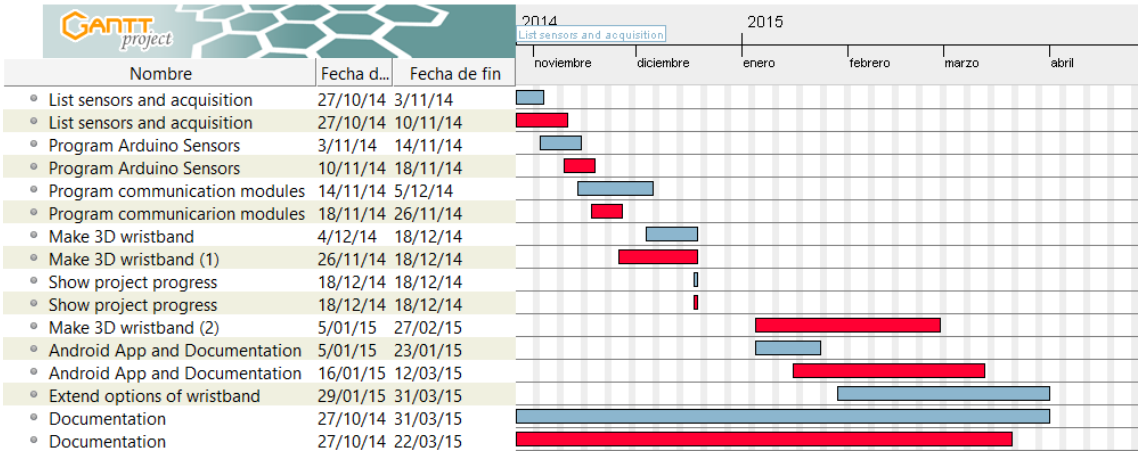


Figure 3: Final Gantt Chart

As it is possible to see on the image above, dates has been changed totally for the entire project. This was caused because some technologies used on the project has changed after the proposal was made, because making more research, it was possible to achieve a cheaper wristband than the other one proposed in October. This is possible to see in the next chapter.

On the other hand, the Android app was a totally issue, because it was needed to learn how to program apps and how to manage the different things wanted to be included on this project, such as SMS or phone calls, apart from the design of the app.

Programming the pulse rate sensor was on schedule, because the information about it was easy to understand and easy to program. Visiting their website, it was possible to get everything to run the pulse rate sensor immediately, just using the C language modules and the documentation downloaded and a few new features, the pulse rate sensor worked properly on those days.

Finally, the most difficult task in this project was the 3D wristband design. This is caused because it was an unknown technology, how to make the design, how to print it, etc. As we can see later, the design of it was totally hard to do it because of the size, the material, etc.

3.2.3. Time management

Every person who talks about their final project, always emphasizes on time management. It was very important to manage this time in order to combine with other modules in this year. This year the module worth 15 ECTS, so 150 hours or above of work on this project.

In this case, time management was very important because some technologies used in this project were unknown, such as Android programming or 3D design. Some hours have been used for learning these technologies, how it works, and how to reach the best way in the project.

Apart from that, coming from another country with another language, some hours of the project have been spent on reviewing the language, trying to correct grammar and expression mistakes.

The image below shows how the time was managed:

Task	Hours estimated	Hours spent
List Sensors and acquisition	5	5
Hardware 1st Prototype	25	25
Hardware 2nd Prototype	5	3
Hardware Final Prototype	2	1
Soldering circuit	6	4
Android app 1st Prototype	40	30
Android app 2nd Prototype	8	8
Android app Final Prototype	4	6
3D Design 1st Prototype	20	25
3D Design 2nd Prototype	12	15
3D Design 3rd Prototype	5	6
3D Design Final Prototype	4	4
Printing Wristband	9	9
Assembling circuit and wristband	5	6
Testing	3	2
Writing Dissertation	30	30
Checking Dissertation	20	10
Total	**203 hours**	**189 hours**

3.2.4. Risk and Issues

In this section it will be explained some of the risks and issues before the project was started. In general, most of these problems came from the Android app and the 3D wristband design. This was analysed before starting with the project and after submitted the project proposal, trying to manage the time between learning and developing the project.

These risks are:

- Android programming is totally unknown. It is necessary to learn about Android programming before start doing anything for the app. On the other hand, Android uses Java, so
- 3D wristband design really difficult. This project should be oriented to a computer science, or information technology project, but 3D was an attractive technology to add on it.
- This project contains big part of electronics, so the knowledge of this field must be more than basics.

4. Design

In this chapter, it is going to be shown how the project will be designed, this means the circuit schematics for the hardware part, some drafts about the Android app interface, and a quick idea about the 3D wristband.

4.1. Hardware

For this project, a device which reads the pulse rate easily and communicates with any Android device in order to interpret the data sent by this device easily will be created.

Apart from that, the device should be the cheapest device made with these purposes, so every element in the circuit will be bought from Chinese retailers, as mentioned before, every component used on the project is open hardware, and so any manufacturer can make these components by its own.

The electronic circuit is very simple; in this case a specific sensor will be used to read the pulse rate on different body parts, but in this project will be on the wrist. Some LEDs will be used in order to see if sensor is working properly; a Bluetooth module will be used to communicate the wristband with any Android device; and finally the main board, an Arduino board will be the main brain of this project. This board is in charge of interpreting any data coming from the sensor and transmit it to the Android device through the Bluetooth module. In addition, this circuit will be power supplied by some button batteries, as much as necessary.

A combination of parallel and serial circuit for these batteries will be made in order to get more voltage than the one given by ant of these batteries. This means that the circuit will have the voltage necessary given by the combination of different parallel circuits connected as a serial circuit. The draft below explains much better how these batteries will work.

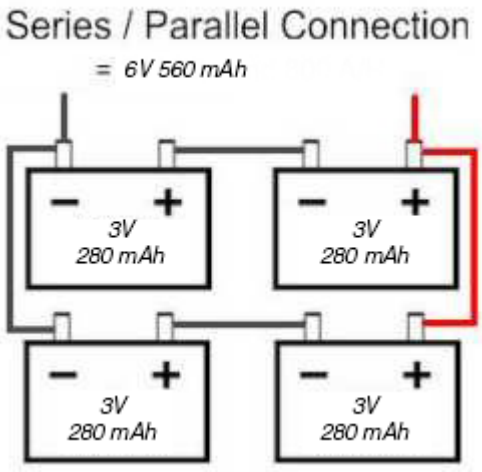


Figure 4: Parallel/Serial Batteries Connection [2]

[2] Image retrieved from (www.accessconnect.com, n.d.)

As seen on the image above, this circuit will have as many batteries as the circuit needs. The parallel connection of some batteries achieves increase batteries capacity. In this case, every parallel circuit has two batteries, so this means that every circuit will have the sum of the batteries capacity, but with the same voltage as before. In order to achieve the voltage needed for the circuit, it is necessary to connect every parallel circuit into a serial circuit. This is necessary because connecting two or more batteries in serial circuit allows us to have the voltage necessary for this circuit. This means a voltage between 3.3V and 5V.

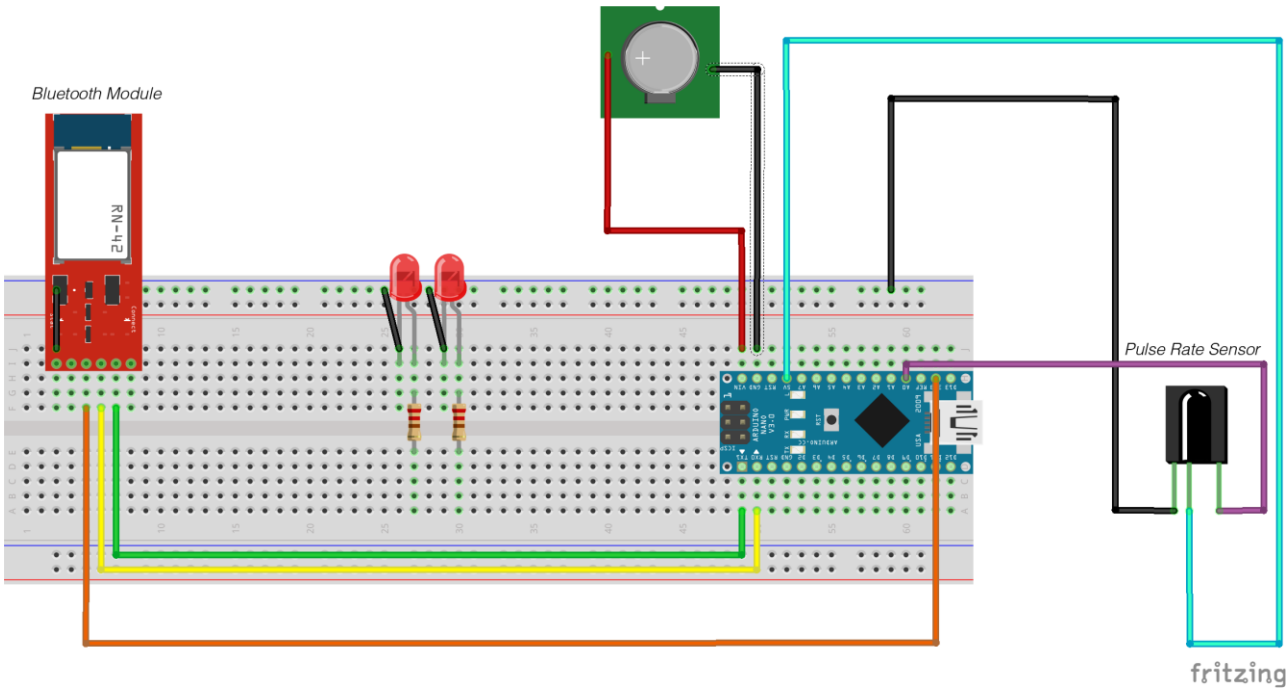


Figure 5: Final Circuit Proposal

As seen on the image above, the final prototype should be like that, because it is the cheapest and easiest way to achieve one of the objectives of this project.

4.2. Android app

4.2.1. Brief introduction

Another project's aim is to make an Android app which interprets correctly the data sent by the sensor. This app must be simple for every user, because this project is oriented in most cases to elderly people, most of them do not know how to use a smartphone correctly or how to setup any app on them.

The app should not require so much information about the user, and should show the main screen as soon as possible when the user opens the app.

4.2.2. Explanation

As mentioned before, the app must be very simple for its use, so each screen will have a short explanation of what is needed to do every moment, in order to guide the user using the app.

First of all, the app will show a splash screen with the app logo. Image below shows a view schematic:

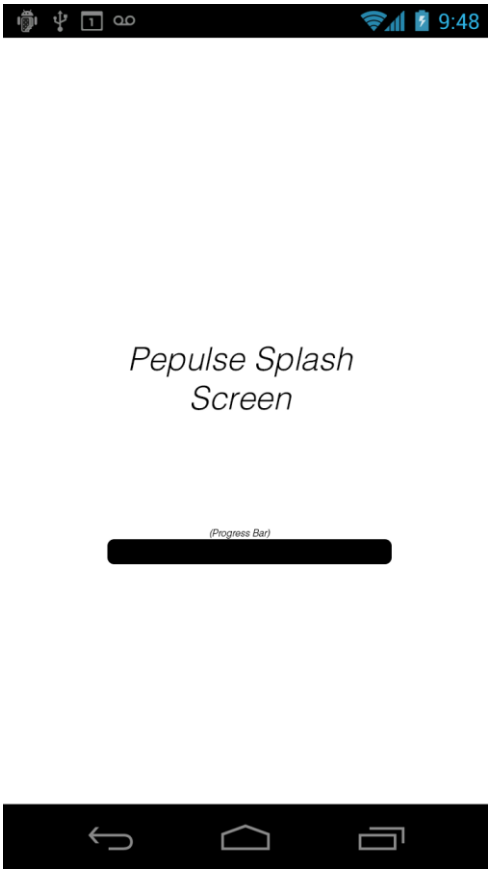


Figure 6: Splash Screen Schematic

Secondly, the app will show a list with every Bluetooth device paired with the mobile phone or tablet. This should be made because it is not properly to put the MAC Address of the Bluetooth device inside the code, and load the app. Each Bluetooth device has its own MAC Address, so that screen it is a must. Images below show a possible view of this screen:

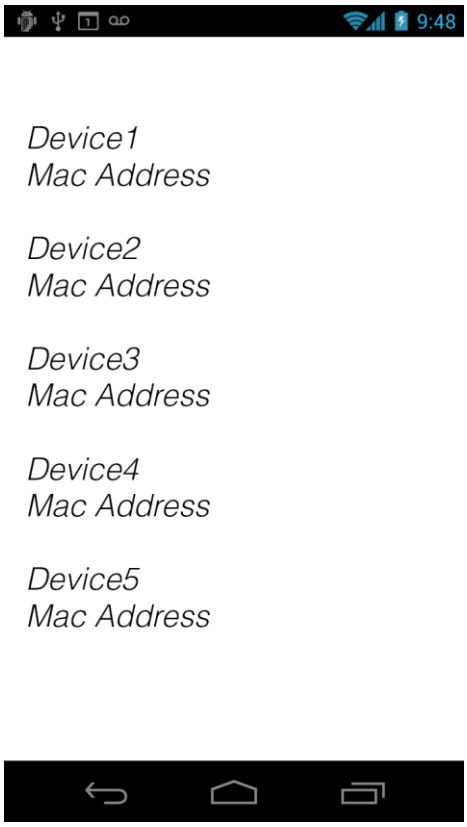


Figure 7: Device List Screen Schematic

Thirdly, the main screen will show the BPM of the user and a button for the app configuration. Clicking that button the app will show the app configuration screen.

Figure 8: Main Screen Schematic

In the configuration screen, the user can introduce any phone number in order to notice someone if the user is in trouble. This is one of the main purposes of the project. Mainly, the app is oriented to elderly people, as mentioned before, so basically if one of the wristband users has a heart-attack, the mobile phone or tablet will notice automatically to the number configured with an SMS with the BPM in order to alarm the mobile number configured by the user. Images below show how the configuration view will be:

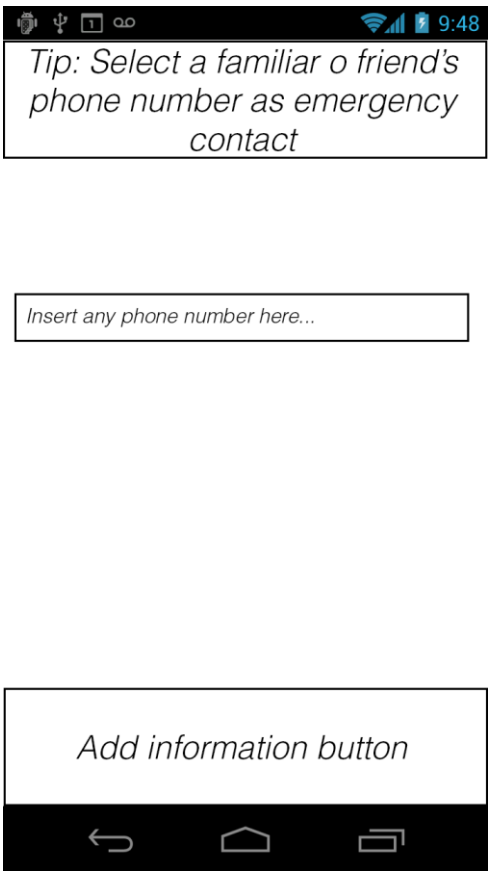


Figure 9: Settings Screen Schematic

The tip on the configuration screen is really important because users could be confused and put their own phone number. So this tip recommend to users to type a different phone number to notice whoever they want.

This is the Android app main idea for this project, but if there is enough time, a database and a login screen will be implemented, in order to register any user who uses this app, and maybe save the data collected by the app and make some statistics of each user.

4.3. 3D Wristband

The last project's objective is to make a personalised wristband for the user. At first time, the best way to fir every component used on this project was to buy a normal

wristband and assembly the components, but in order to use another technology, the idea of using 3D printers was really good.

At the beginning it was a difficult task to think how to make this wristband in 3D, because it is really hard to imagine how a wristband can be made, from nothing. A researching was made about different models of wristbands in order to learn how to make one. Google Sketchup was the software used for this purposes, which includes a specific tool to look for every model the users have made and upload to this platform. Wristbands are not very famous for 3D designers; there were a couple of them uploaded on this platform. Therefore, the wristband will be made from zero and the main idea is:

Figure 10: Wristband [3]

This image above shows the perfect wristband for this project, because it is possible to assembly every board on it without any size problem. In addition, the idea is to print this wristband with flexible material, so the wristband size does not matter at all. In the case the wristband will be printed in rigid plastic, the design will change completely, because it will be impossible for the users to fit the wristband showed above in their wrists, because this material is impossible to bend.

[3] Image retrieved from (www.batag.com, n.d.)

5. Implementation

In this section, the different steps followed, prototypes and models has been made for the project are going to be shown. This means every single module used for the wristband, explaining what they are, their functionality and how they are combined to make the entire system. Apart from that, the most relevant parts of the code are going to be shown in order to explain what it is happening and how the pulse sensor works.

On the other hand, the different things the app has, how it works and its code to explain what is happening will be explained.

At last, the different wristbands made in 3D will be shown, because it was not an easy task to accomplish and needs to be explained every step it was made and why.

Also, there will be some photographs of different prototypes made for this project.

5.1. Brief introduction

As mentioned before, this project has different parts, such as hardware, software and 3D model.

The hardware part will consist on the elements the wristband has; the boards used, sensors and other elements.

The software part will consist on a simple Android app to read the heart rate, and if the user has any problem with its heart, notice someone the user configured at the beginning to warn that the user is in trouble.

The 3D model consists on the wristband itself, it will be used to assembly each board, LEDs and the batteries.

5.2. Hardware

5.2.1. 1st prototype

In this prototype, the most important thing was to assembly everything and sees how it works. Apart from that, this was the first time using a pulse sensor rate, so it was important to know how to read the data the pulse sensor sends and how to manage it.

Figure 11: Bluetooth Module

In the photo above is possible to see the Bluetooth module will be used in this project. This module will communicate the wristband with any Android device which has the app installed. At the beginning, sends data constantly, so that would be a waste of batteries. The Bluetooth module has 4 pins, the first two make the communication between the module itself and the Arduino board [TX (Transmitter) and RX (Receiver)]; last two pins are used to connect to the power supply. In this case, the Arduino board will power supply each component.

Figure 12: Pulse Rate Sensor

In this photo is possible to see the sensor is going to be used. It has an infrared led in the centre of it, which can read the pulse rate perfectly. This sensor consists of 3 pin to connect to the board. The right one is used to send the data, the remaining pins are used to connect to the power supply, as the Bluetooth module, to the Arduino board.

Figure 13: LEDs and Resistors

These are the LEDs (light-emitting diodes) and its resistors. Its function is basically shows if the sensor is working properly and if it is taking the pulse rate correctly. The resistors are used to maintain the voltage given to the LEDs, because if the voltage is higher can break the LEDs. The longest pins on LEDs are the anodes, which combined with the resistors, will be connected to each pin necessary on the Arduino board. The other two pins can be combined on the breadboard in order to collect every ground (a.k.a. GND) connection.

Figure 14: Arduino Uno

This is the Arduino board used in the first prototype. It is the hardware's main brain, it is in charge to collect all data from the sensor and send it through the Bluetooth module. At the beginning, the USB cable was being used as a power supply, but it was impossible to connect the board to the laptop via Bluetooth if the board is connected though the USB cable, so some batteries were used in order to change this and make the circuit work properly with the Bluetooth module.

Figure 15: Batteries

This was used as a power supply for the entire project. Consists of 4 batteries AAA type and a Jack connector soldered with two cables. Those batteries give 5V, and connected as a serial circuit has four times the capacity of one battery.

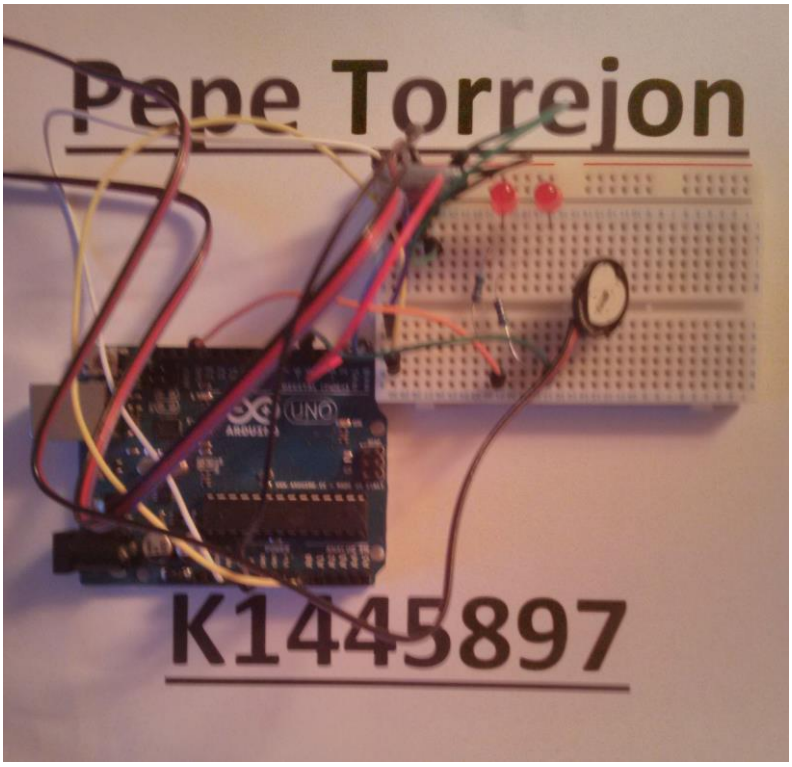


Figure 16: Circuit Assembled

This photo shows the first prototype. Testing was successful connecting the Arduino to the laptop though Bluetooth.

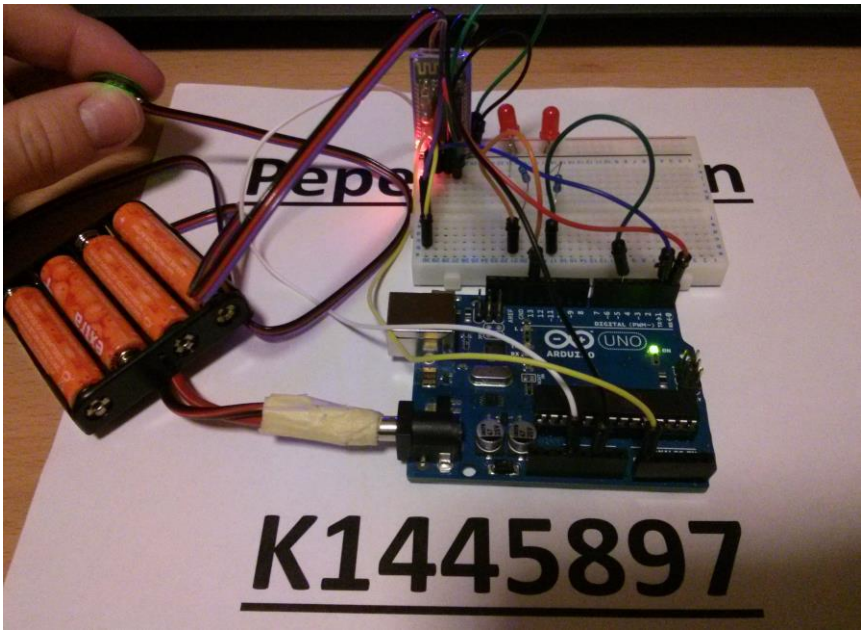


Figure 17: Circuit Working

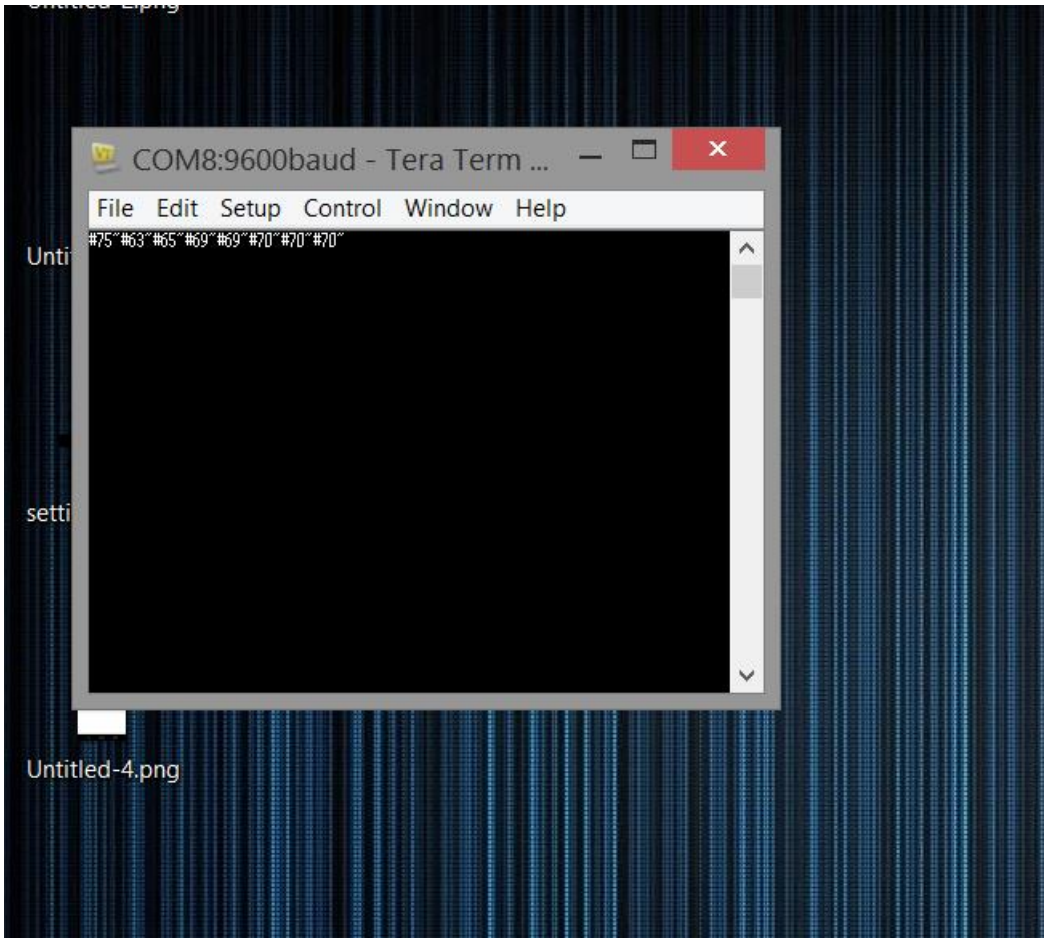


Figure 18: User's BPM

Some photos of the testing. The first one shows the project turned on, LEDs working and one of the body parts used to read the pulse rate. The second photo shows the information gathered by the laptop when it is connected to the circuit. Just only shows the pulse rate (BPM) when the sensor is closer to any part of the body where it is possible to read the pulse rate.

5.2.2. 2nd prototype

In this prototype, the main board was changed to another because of its size. This new board has the same specifications as the old one, so only changed the size of the project, smaller than before.

Figure 19: Arduino Nano

This is the Arduino Nano board; this one has the same specifications as the one which was used before. The only difference between both of them is the size, as we can see on the photo above.

This prototype is more oriented to be on the wristband, and also it would be the final prototype but in the middle of the project it was discovered other Arduino board smaller than this one, included on the final prototype.

5.2.3. Final Prototype

This is the circuit will be assembled on the wristband, the one definitive. It is only change the main board, to an Arduino Micro. In the photo below it is possible to see the size change.

Figure 20: Arduino Nano vs Arduino Pro Mini

The other things are the same, so the final arranges to do were put these boards on the wristband and solder them. To end the circuit, it was necessary to solder everything and try it, so in the image below is possible to see the whole circuit soldered and stuck over the wristband.

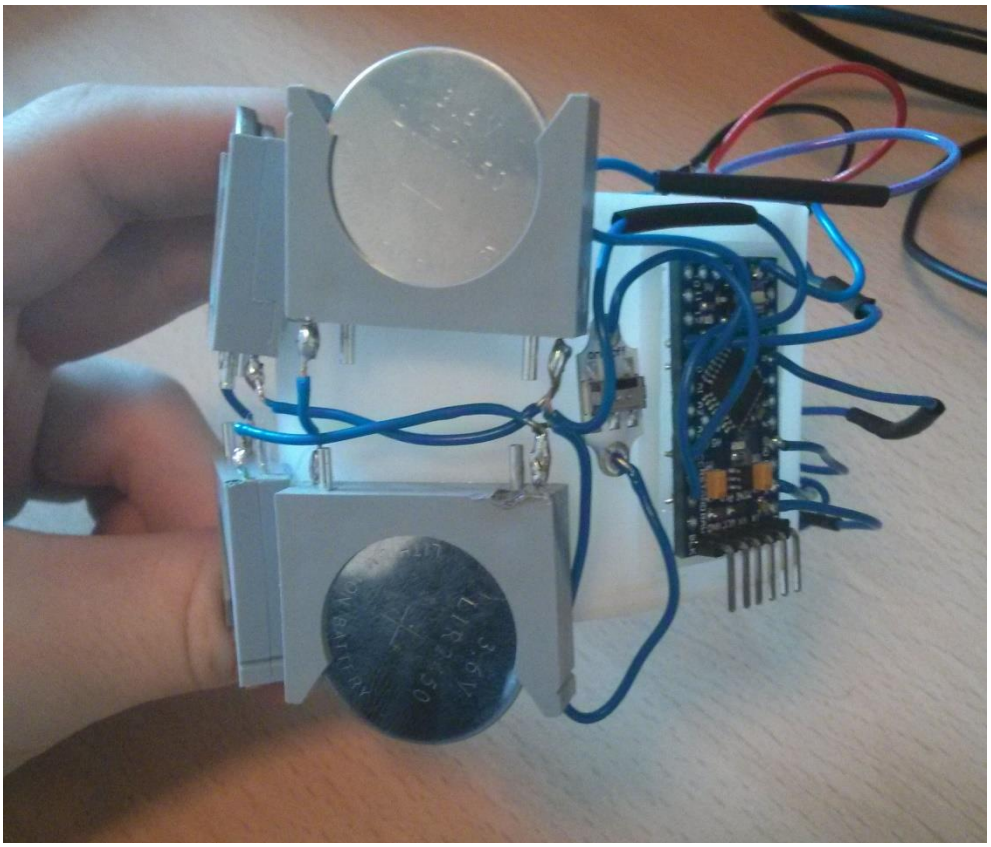


Figure 21: Batteries, switch and Arduino Pro Mini

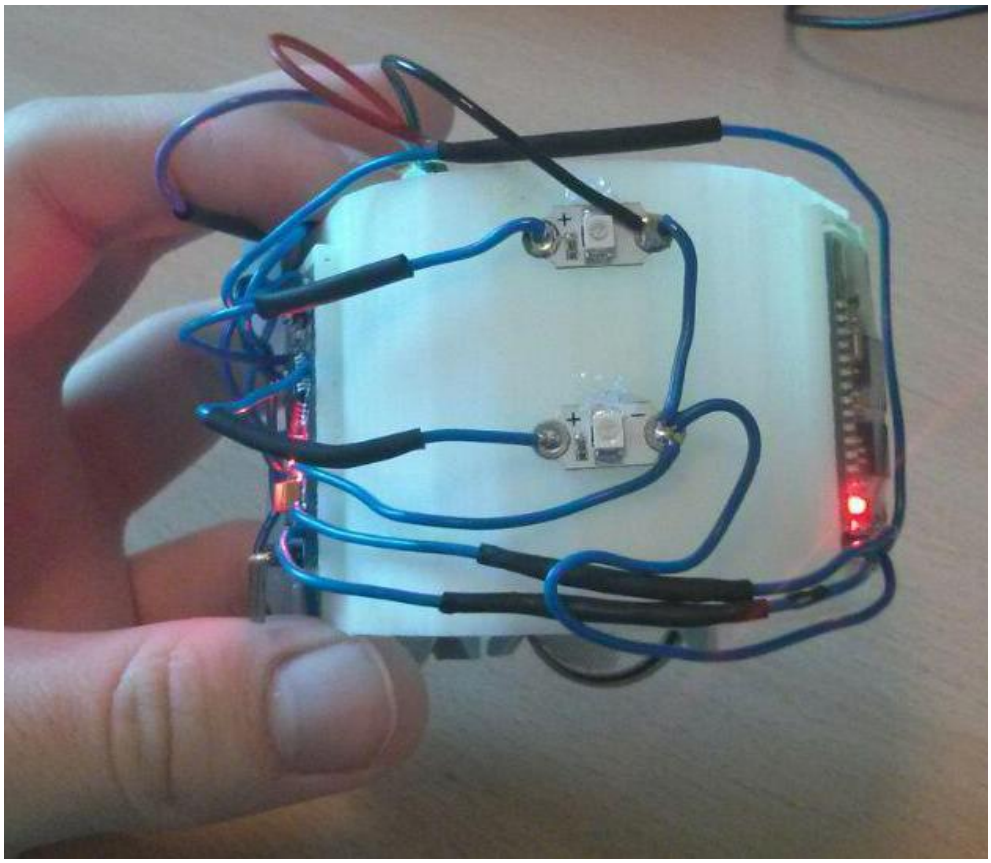


Figure 22: LEDs and Bluetooth module

Finally, some batteries with more than 3V were found, so the batteries circuit finally changed into a parallel circuit only, getting from it the capacity sum of the batteries used. The image below shows how the batteries used were finally connected:

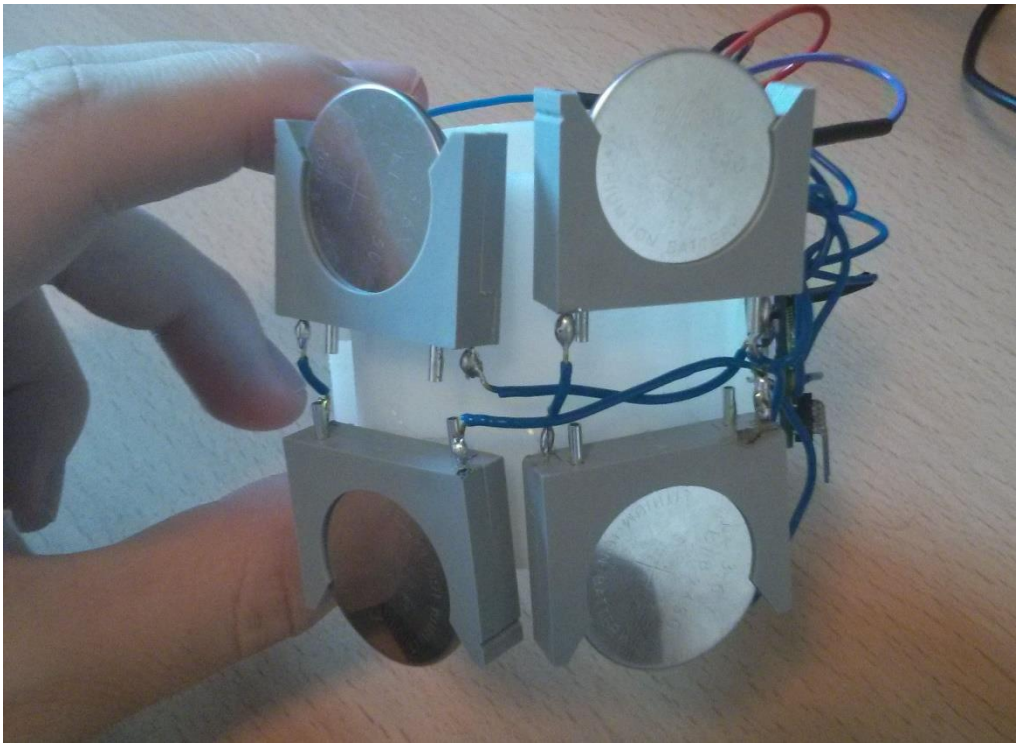


Figure 23: Batteries LIR2450 3.6v

Although these batteries were found, the serial/parallel circuit had to be done because with only parallel circuit running was not giving the enough power supply for each component finally.

5.2.4. Arduino Code

The Arduino board was programmed in C, the language support by these boards. In this case, the sensor uses a specific C library hosted in the official website. The code the manufacturer provides is made to communicate with **Processing** IDE. **Processing** is a programming language and an IDE based on Java, and is used in most cases for multimedia purposes. The manufacturer provides both Arduino library and Processing codes, and the purpose of it is to convert the PC in a heart monitor. For this project the code was almost useless, only the C library made for the sensor was useful, used it on this project as well. The serial communication was achieved thanks to the Arduino library provided on the website combined with a piece of code for the Arduino board.

```
void setup(){
  pinMode(blinkPin,OUTPUT);         // pin that will blink to your heartbeat!
  pinMode(fadePin,OUTPUT);          // pin that will fade to your heartbeat!
  Serial.begin(9600);             // we agree to talk fast!
  interruptSetup();                 // sets up to read Pulse Sensor signal every 2mS
}

void loop(){

  //Send 'Signal' to any Android Device connected via Bluetooth
  if(signal_flag == true){

     fadeRate = 255; // Set 'fadeRate' Variable to 255 to fade LED with pulse
     if(Serial.available()){blueToothVal = Serial.read();} //If there's any Serial device connected, read the Serial flow

        Serial.println("#");
        Serial.println(BPM); //Print the pulse rate
        Serial.println("~");

     signal_flag = false;//Reset the flag
  }
  ledFadeToBeat();
  delay(20);                             //Just to make some time between each action
}


void ledFadeToBeat(){
    fadeRate -= 15;                         //  Set the fade value for LEDs
    fadeRate = constrain(fadeRate,0,255);   //  Constrain makes any number to be in that value range
    analogWrite(fadePin,fadeRate);          //  Blink Led
  }
```

Figure 24: Arduino Code

The code above is used in the Arduino board. The **setup()** and **loop()** methods must be in every piece of code for Arduino board. The **setup()** method just tell to the board how to manage their digital and analog pins, how to manage the dataflow in those pins and the Serial communication speed. The **loop()** method is used by these boards as an infinite loop, the piece of code inside that method starts over and over again, until the board is turned off.

Inside the **loop()** method, it is possible to see how the Serial communication is managed. In this case, the BPM is being read constantly, and sent through the serial port. The BPM is sent between "#" and "~" symbols in order to differentiate each user's BPM

5.3. Software

5.3.1. 1st Prototype

In this prototype, the main objectives to be achieved for the app are to show a splash screen, to load the different Bluetooth devices connected to the mobile phone and to show only the pulse rate.

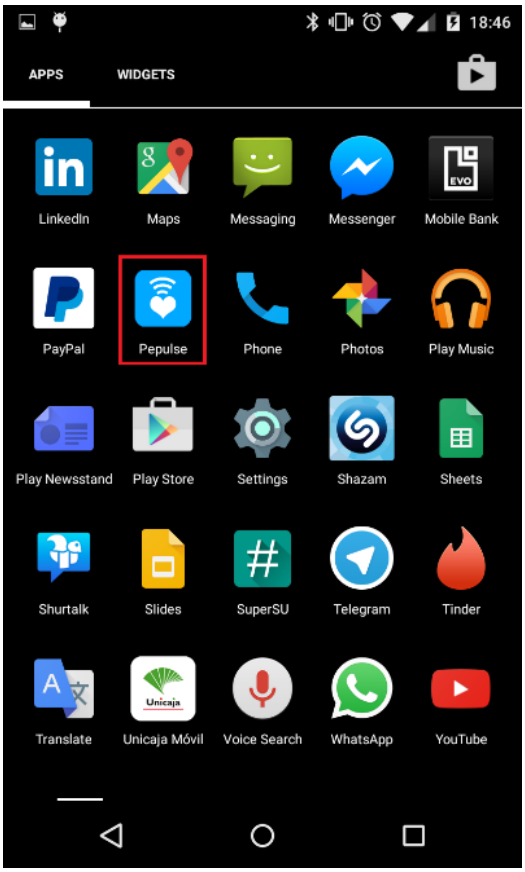


Figure 25: Pepulse Icon

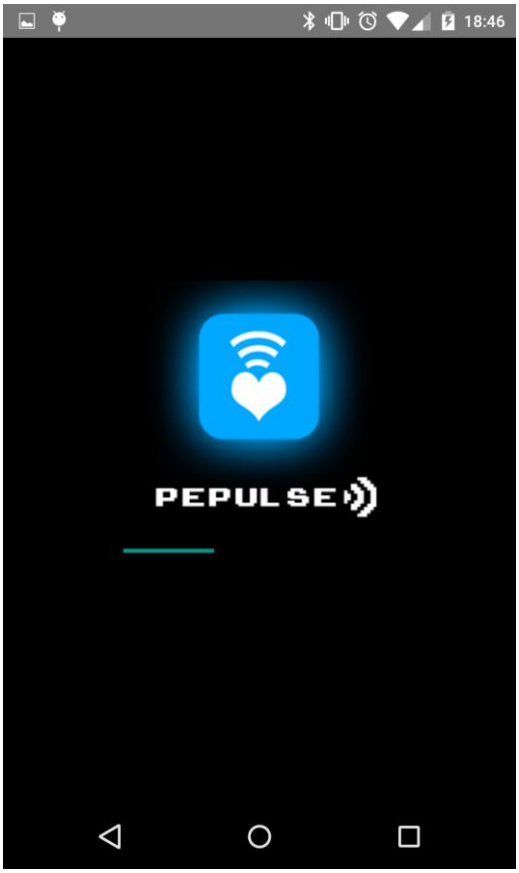


Figure 26: Pepulse Splash Screen 1st Prototype

The images above show the app icon and the splash screen made with Adobe Photoshop CS6 used for the 1st prototype.

Figure 27: Splash Screen XML File

The most relevant things are the different properties about the background in the splash screen, how the app must configure the screen to fit in each device, and the different elements tag used to load everything correctly in the splash screen.

The <LinearLayout> tag creates a Linear Layout inside the screen. This means that everything inside this Layout are stacked one by one, vertically or horizontally. Reminder that Android requires one layout minimum to include everything we want to see in the screen. The properties used for the layout ("**fill_parent**" and "**wrap_content**" indicates how the layout will fit in the screen. In this case the width will be the same as the phone screen using the app, and the height will be the same as the sum of the elements inside of it.

The <ImageView> tag is used to load images and makes the app show them. In this case the image showed is the icon made for this splash screen. The most relevant things are the "**android:id**" property, which gives an id for this tag (it will be explained later why we need to use this), and the "**android:background**" property, which loads the image we want.

Finally, the <ProgressBar> tag is used to make a progress bar for the splash screen and show on it.

After all, it is necessary to combine this XML piece of code with a Java class, in order to give functionality to this screen, for example the progress bar animation.

```
import ...
/**
 * Created by pepe on 22/01/15.
 */
public class SplashScreenActivity extends Activity {
    private static final int seconds = 5;
    private static final int miliseconds = seconds*1000;
    private static final int delay = 2;
    private ProgressBar pbar;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.splash_screen);
        pbar = (ProgressBar) findViewById(R.id.splash_bar);
        pbar.setMax(maxProgress());
        startAnimation();
    }
    public void startAnimation(){
        new CountDownTimer(miliseconds,1000){
            @Override
            public void onTick(long millisUntilFinished) {
                pbar.setProgress(establish_progress(millisUntilFinished));
            }
            @Override
            public void onFinish() {
                Intent i = new Intent(SplashScreenActivity.this, DeviceListActivity.class);
                startActivity(i);
                finish();
            }
        }.start();
    }
    public int establish_progress(long millis){
        return (int)(miliseconds - millis)/1000;
    }
    public int maxProgress(){
       return seconds-delay;
    }
}
```

Figure 28: Splash Screen Java Class

As always, a Java class must be created to include every piece of code we want to run. In this case, a Java class will be created per XML file we have. As seen in the image above, the method **onCreate()** specifies what objects are the screen going to show. Some examples of what is shown on the screen are the method **setContentView(R.layout.splash_screen)**, indicates which layout must show on the screen; **findViewById(R.id.splash_bar)**, is used to find the progress bar created in the XML file and save it in a variable inside the Java class. Then, this progress bar is used to make an animation of itself when the app is loading.

The rest of the code is for making the progress bar animation. The method **startAnimation()** just makes the animation start. The behaviour of it is inside of this method; in this case, the progress bar was made with a countdown timer, because it is impossible link the modules the app are loading with the progress bar.

Finally, the method **onFinish()** inside the **startAnimation()** method tell us what is going to do after the progress bar finishes. In this case it is going to show us the Bluetooth devices list paired with the mobile phone using the app. The image below show how the screen is and how manages these Bluetooth devices:

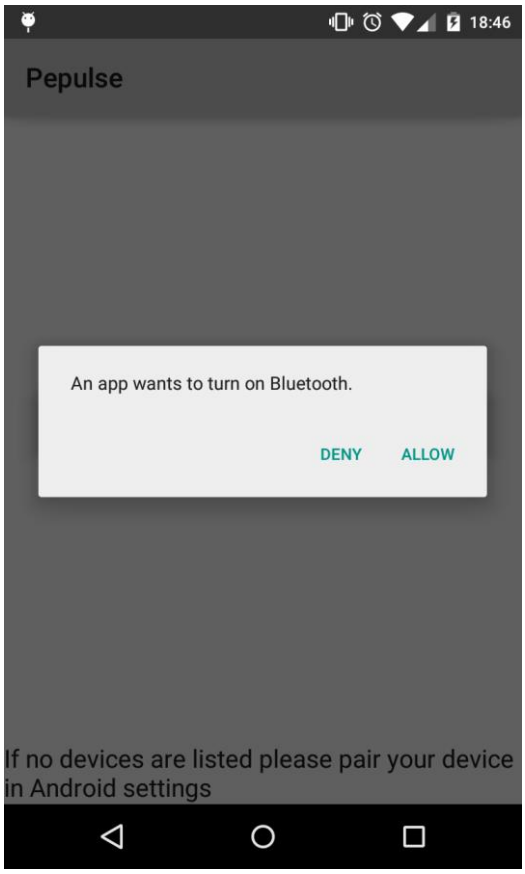


Figure 29: Bluetooth Request

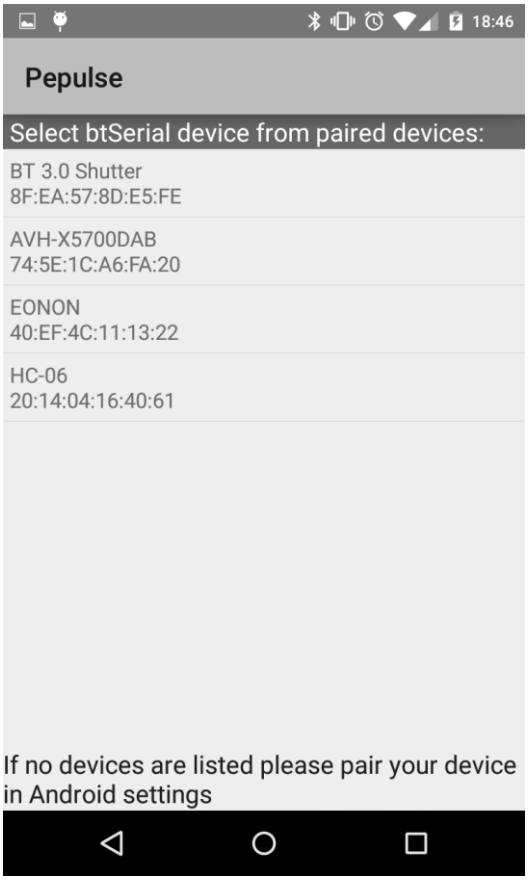


Figure 30: Device list screen 1st Prototype

The one called ‘HC-06’ is our wristband, which is the Bluetooth module of the circuit.

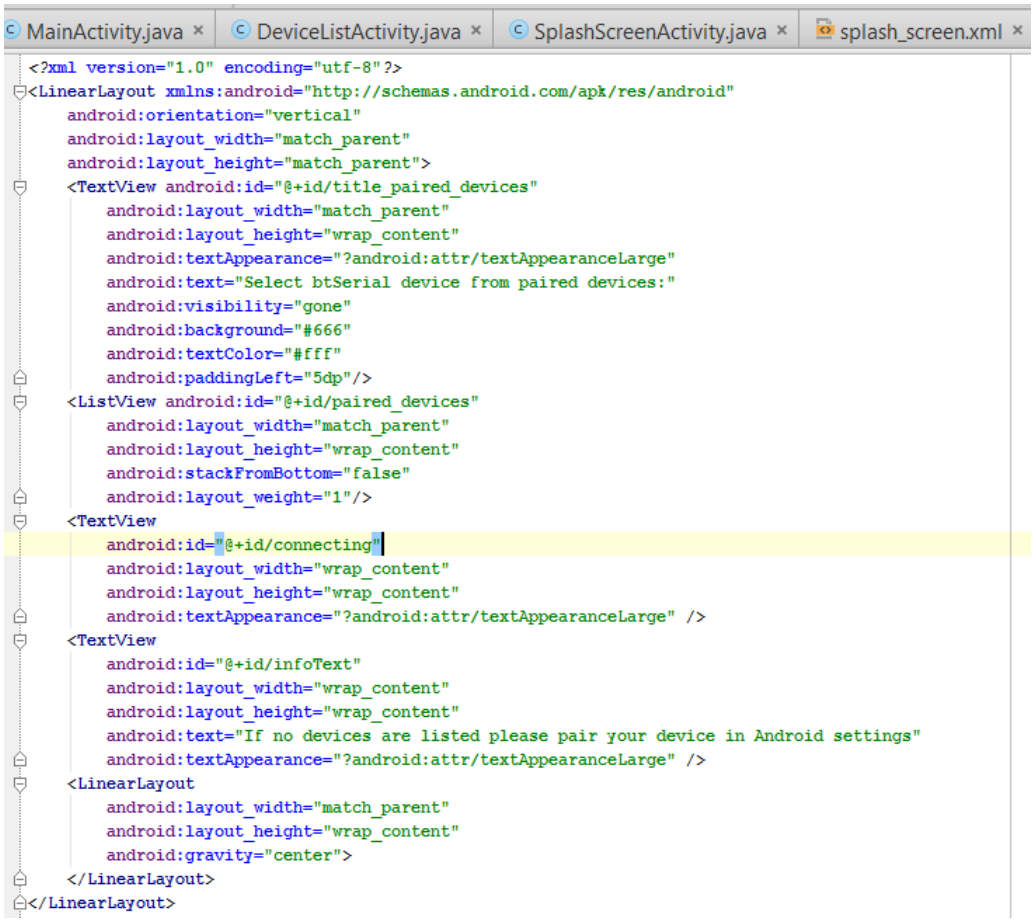


Figure 31: Device list screen XML File

This XML code is not too much different than the shown before, some tags are new here, like <**TextView**> and <**ListView**>. <**TextView**> tag shows any text we want to show. As seen before in the XML code image, the first **<TextView>** shows the text “**Select btSerial device from paired devices**”. On the other hand, <**ListView**> tag shows, in this case, every Bluetooth device paired with the mobile phone using the app.

Afterwards, the most relevant parts from the Java code will be shown, because it is so long to copy here and explain it:

```
@Override
public void onResume()
{
    super.onResume();
    checkBTState(); //Check the BT status, if it is off, asks the user to turn it on

    textView1 = (TextView) findViewById(R.id.connecting);
    textView1.setTextSize(40);
    textView1.setText(" ");

    //Initialize ArrayAdapter for devices linked in the actual mobile phone
    mPairedDevicesArrayAdapter = new ArrayAdapter<String>(this, R.layout.device_name);

    //Set up the list for paired devices
    ListView pairedListView = (ListView) findViewById(R.id.paired_devices);
    pairedListView.setAdapter(mPairedDevicesArrayAdapter);
    pairedListView.setOnItemClickListener(mDeviceClickListener);

    //Get the BT module from the mobile phone
    mBtAdapter = BluetoothAdapter.getDefaultAdapter();

    //Get the devices linked in the mobile phone
    Set<BluetoothDevice> pairedDevices = mBtAdapter.getBondedDevices();

    //And show them on the list
    if (pairedDevices.size() > 0) {
        findViewById(R.id.title_paired_devices).setVisibility(View.VISIBLE);
        for (BluetoothDevice device : pairedDevices) {
            mPairedDevicesArrayAdapter.add(device.getName() + "\n" + device.getAddress());
        }
    }
}
```

Figure 32: Device list screen onResume() method

This method **onResume()** is useful either the app is ran for the first time or in any moment it is needed to come back to this screen once you have accessed to the app. This means that you can select multiple Bluetooth devices, one per use, in any moment the user is running the app. **onResume()** method makes this screen accessible whenever the user wants. Inside of it, the devices list is configured in order to make it nicer for the user. The different paired devices in the mobile phone will be shown on the list with their MAC Addresses, so the user will not have any problem to locate the wristband.

```
private OnItemClickListener mDeviceClickListener = (av, v, arg2, arg3) -> {

        textView1.setText("Connecting...");
        // Get the device MAC address, which is the last 17 chars in the View
        String info = ((TextView) v).getText().toString();
        String address = info.substring(info.length() - 17);

        // Make an intent to start next activity while taking an extra which is the MAC address.
        Intent i = new Intent(DeviceListActivity.this, MainActivity.class);
        i.putExtra(EXTRA_DEVICE_ADDRESS, address);
        startActivity(i);
};
```

Figure 33: Device list screen OnItemClickListener() method

When the user clicks on any Bluetooth device from the list, the method **OnItemClickListener()** takes action. In this case, when the user push over the

wristband on the devices list, a little message appears while the app is trying to connect to the Bluetooth device. Once the phone is connected, the **startActivity()** method changes the screen to the main one. The **startActivity()** method has an **Intent** object, which is composed of the screen is using the **Intent** we have created, and the screen we want to show once the actual screen is finished and the **startActivity()** method is launched.

```
private void checkBTState() {
    // Check device has Bluetooth and that it is turned on
    mBtAdapter=BluetoothAdapter.getDefaultAdapter();
    if(mBtAdapter==null) {
        Toast.makeText(getBaseContext(), "Device does not support Bluetooth", Toast.LENGTH_SHORT).show();
    } else {
        if (mBtAdapter.isEnabled()) {
            Log.d(TAG, "...Bluetooth ON...");
        } else {
            //Prompt user to turn on Bluetooth
            Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableBtIntent, 1);

        }
    }
}
```

Figure 34: Device list screen checkBTState() method

This method check if the Bluetooth is turned on. If the user's mobile phone does not have Bluetooth, the app will throw and error advertising that the mobile phone the user has does not have Bluetooth. In the case the Bluetooth is turned off, the app asks the user to turn it on.

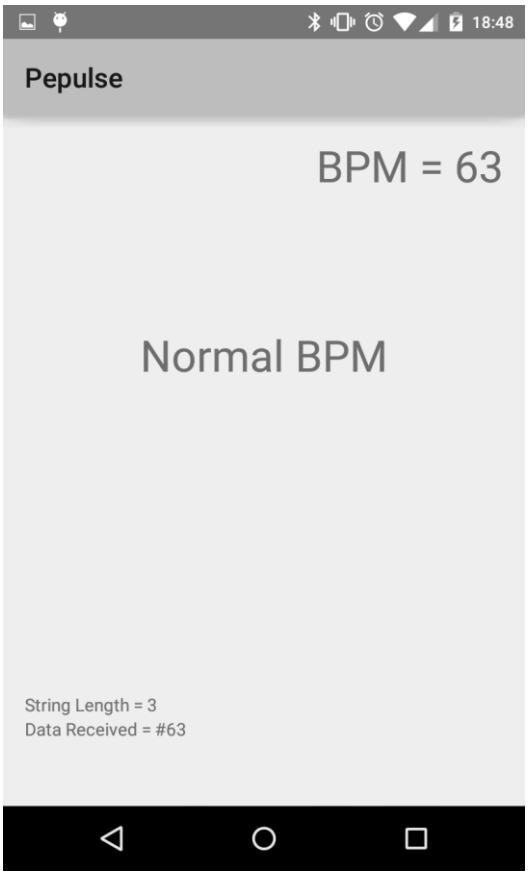


Figure 35: Main screen 1st prototype

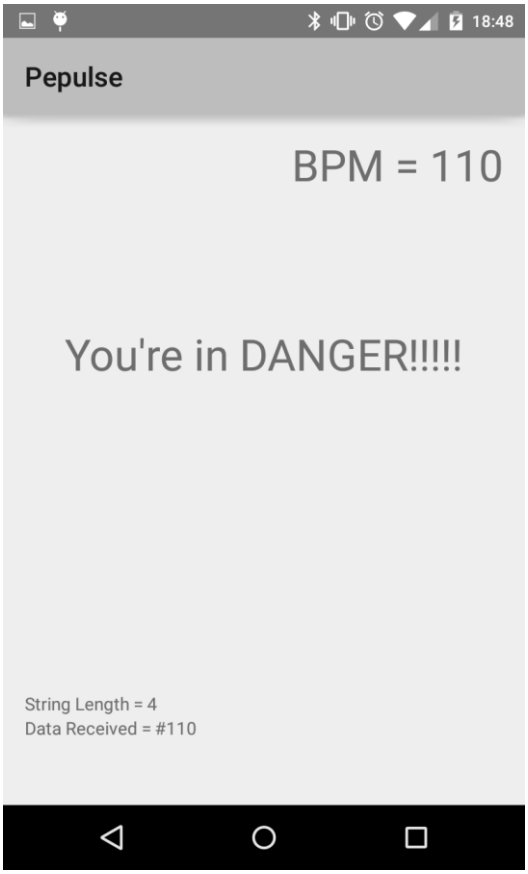


Figure 36: Main screen 1st prototype "Danger"

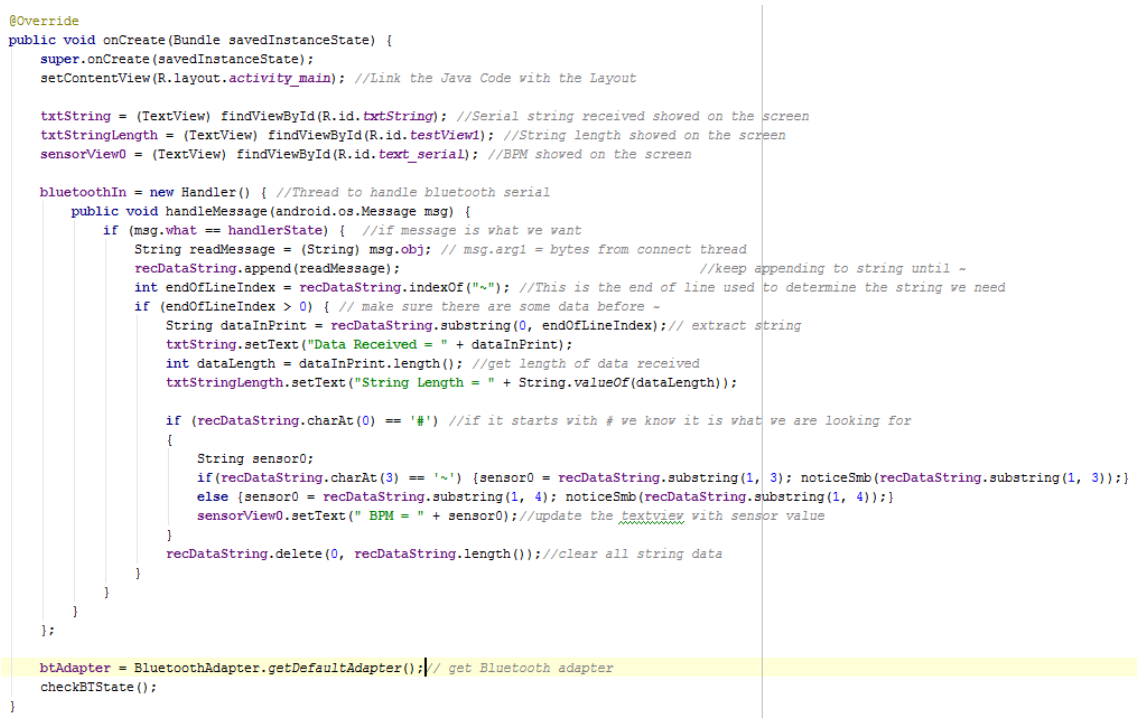
In this screen, it is shown the BPM taken by the sensor. It is updated every moment it takes pulse from the wrist.

MainActivity.java × DeviceListActivity.java × activity_main.xml ×

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent" android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin" tools:context=".MainActivity">
    <TextView
        android:id="@+id/testView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_above="@+id/txtString"
        android:text=""
        android:textSize="15sp" />
    <TextView
        android:id="@+id/txtString"
        android:layout_width="wrap_content"
        android:layout_height="50dp"
        android:layout_alignLeft="@+id/testView1"
        android:layout_alignParentBottom="true"
        android:text=""
        android:textSize="15sp" />
    <TextView
        android:id="@+id/text_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="140dp"
        android:text=""
        android:textSize="32dp"
        android:textAppearance="?android:attr/textAppearanceMedium" />
    <TextView
        android:id="@+id/text_serial"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerHorizontal="true"
        android:layout_alignParentRight="true"
        android:text="BPM = ????"
        android:textSize="32dp"
        android:textAppearance="?android:attr/textAppearanceMedium" />
</RelativeLayout>
```

Figure 37: Main screen XML File

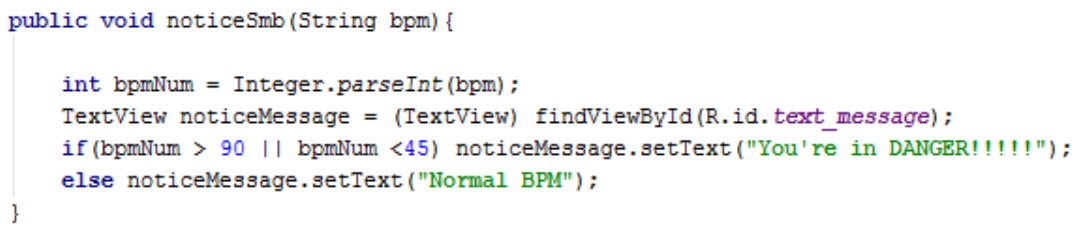
In the code above, the different elements used on the main screen are defined. In this case a **RelativeLayout** is used, because this layout allows the developers to locate every element used on the screen wherever they want, totally different to the LinearLayout. Remind this layout locate every element one after the other. As seen in the image above, each **TextView** is used for different purposes, but the important one is the one called **text_serial** in the **android:id** property. This one will show us the user's BPM.



```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main); //Link the Java Code with the Layout

    txtString = (TextView) findViewById(R.id.txtString); //Serial string received showed on the screen
    txtStringLength = (TextView) findViewById(R.id.testView1); //String length showed on the screen
    sensorView0 = (TextView) findViewById(R.id.text_serial); //BPM showed on the screen

    bluetoothIn = new Handler() { //Thread to handle bluetooth serial
        public void handleMessage(android.os.Message msg) {
            if (msg.what == handlerState) {  //if message is what we want
                String readMessage = (String) msg.obj; // msg.arg1 = bytes from connect thread
                recDataString.append(readMessage);                                      //keep appending to string until ~
                int endOfLineIndex = recDataString.indexOf("~"); //This is the end of line used to determine the string we need
                if (endOfLineIndex > 0) { // make sure there are some data before ~
                    String dataInPrint = recDataString.substring(0, endOfLineIndex);// extract string
                    txtString.setText("Data Received = " + dataInPrint);
                    int dataLength = dataInPrint.length(); //get length of data received
                    txtStringLength.setText("String Length = " + String.valueOf(dataLength));

                    if (recDataString.charAt(0) == '#') //if it starts with # we know it is what we are looking for
                    {
                        String sensor0;
                        if(recDataString.charAt(3) == '~') {sensor0 = recDataString.substring(1, 3); noticeSmb(recDataString.substring(1, 3));}
                        else {sensor0 = recDataString.substring(1, 4); noticeSmb(recDataString.substring(1, 4));}
                        sensorView0.setText(" BPM = " + sensor0);//update the textview with sensor value
                    }
                    recDataString.delete(0, recDataString.length());//clear all string data
                }
            }
        }
    };

    btAdapter = BluetoothAdapter.getDefaultAdapter();// get Bluetooth adapter
    checkBTState();
}
```

Figure 38: Main screen onCreate() method

One of the most relevant methods of the main screen is the **onCreate()** method. This method, apart from what has been described before, in this case this method is used to get the data sent by the wristband, and manage it. It was necessary to create a thread apart from the main one to handle this messages. For debugging, it is used some TextViews in order to show what we receive from the wristband. Once we have seen what we received, it was easier to manage the data received. It was really necessary to include some symbols to distinguish between BPMs received from the wristband.



```
public void noticeSmb(String bpm){

    int bpmNum = Integer.parseInt(bpm);
    TextView noticeMessage = (TextView) findViewById(R.id.text_message);
    if(bpmNum > 90 || bpmNum <45) noticeMessage.setText("You're in DANGER!!!!!");
    else noticeMessage.setText("Normal BPM");
}
```

Figure 39: Main screen SMS method

Another relevant method for the main screen is the one which notices somebody if the user is in trouble, because the BPM is too high or another reason. For the first prototype, the use of this method was to change a **TextView** to check if the app would respond correctly.

5.3.2. 2nd Prototype

Using the first prototype, the second prototype was made, changing a few things of the first one, and adding more options the app needs to achieve every objective mentioned at the beginning.

Firstly, the app appearance has been changed, for example the options button and the Bluetooth list devices, in order to show a nicer view of them.

The splash screen also has been changed. The reason why it was changed it is because actually apps have this type of design, with “plane colours” and very simple. In addition, the purpose of this design, as the other designs like this, is to highlight the main thing of the app. In this case, the first thing we want users to see is their pulse rate BPM. The image below shows the final splash screen:

Figure 40: Splash Screen 2nd Prototype

In addition of the new splash screen logo, a design for the progress bar has been added, in order to make it with the same colour as the icon above. Apart from that, the code for splash screen is still the same, changing colours and the icon. It is possible to see in the image below:

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#00a8ff">

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center">

        <ImageView
            android:layout_width="423px"
            android:layout_height="640px"
            android:id="@+id/splash_image"
            android:layout_gravity="center_horizontal"
            android:background="@drawable/splash" />

        <ProgressBar
            style="?android:attr/progressBarStyleHorizontal"
            android:layout_width="204dp"
            android:layout_height="wrap_content"
            android:id="@+id/splash_bar"
            android:layout_gravity="center_horizontal"
            android:mirrorForRtl="false"
            android:nestedScrollingEnabled="false"
            android:progressDrawable="@drawable/progress_bar" />

    </LinearLayout>
</LinearLayout>
```

Figure 41: Splash screen 2nd Prototype XML File

As seen before, the **<ImageView>** tag is still loading the image in the centre of the splash screen. In order to get the whole screen with the same colour as the image's background color, the property "**android:background="#00a8ff**" was added, same as the first prototype, but in this case the background image colour changed. Apart from that, the progress bar has been changed too, in order to make it with the same style as the image and the rest of the app design.

Also, the Bluetooth devices list view has been changed with the same design, same colours and adding some padding between elements on the list. The image below show how it was made:

Figure 42: Device list screen 2nd prototype

Comparing to the code before, this has been changed in a few properties of the list and the two texts on the view. Parameters like the padding and margins has been added in order to make this view nicer. In the code below it is possible to see the new features of these elements:

```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:textSize="18sp"
    android:padding="5dp">

</TextView>
```

Figure 43: Elements in device list screen

This is the **TextView** corresponding to each element in the device list. As mentioned before, in the image above is possible to see the **textSize** and **padding** properties have been added in order to achieve a nicer view of the device list.

The main view of this app also has been changed, in this case the BPM number is the first thing the user can see, because the number size is really big, achieving one of the objectives of this type of design. The image below shows the final design of this view:

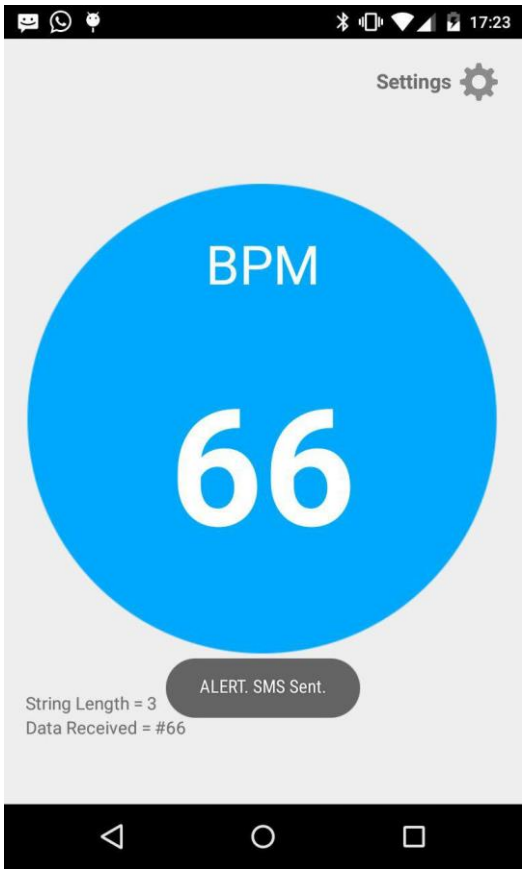


Figure 44: Main screen 2nd prototype

```
<ImageView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/imageView"
    android:layout_centerVertical="true"
    android:layout_alignLeft="@+id/textView"
    android:layout_alignStart="@+id/textView"
    android:src="@drawable/circle" />
```

Figure 45: Main screen blue circle

This **ImageView** object is the circle seen on the main screen, the most relevant thing is the **android:src** property, which loads the image wanted to be shown on this screen, in this case the blue circle in the middle.

```
<TextView
    android:id="@+id/text_serial"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="000"
    android:textSize="120dp"
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true"
    android:textStyle="bold"
    android:layout_marginTop="-60dp"
    android:paddingTop="60dp"
    android:textColor="#fff" />

<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="BPM"
    android:id="@+id/textView"
    android:textSize="40dp"
    android:layout_above="@+id/text_serial"
    android:layout_centerHorizontal="true"
    android:layout_marginBottom="40dp"
    android:textColor="#fff" />
```

Figure 46: Main screen user's BPM

These two **TextViews** are the BPM seen above the blue circle. As seen before in the first prototype, makes the same thing but some new properties like the **android:textColor**, to change the text color; or **android:paddingTop** to make some margin between those two texts, appear in this prototype.

```
<ImageView
    android:layout_width="64px"
    android:layout_height="64px"
    android:id="@+id/imageView2"
    android:layout_gravity="right"
    android:layout_alignParentEnd="true"
    android:src="@drawable/settings"
    android:clickable="true" />
```

Figure 47: Main screen – Settings button

The last relevant thing in this part is the settings button. In this case, if the image is clicked by the user, the settings screen appear. That is the reason why the property **android:clickable** appears, making any image clickable if it is needed. As seen before, the property **android:src** loads the image from the **drawable** folder inside the **Android project**.

Finally, in this prototype was included the app settings panel. This app is able to save any phone number introduced by the user, and send an SMS with user statistics to

the phone number saved. This project, as mentioned on the proposal, is oriented mostly to elderly people but also people with heart problems. If suddenly any of these people have a problem and they are not capable to notice someone while they are having a heart attack or something similar, the mobile phone will do automatically. The settings panel is only composed with a text field which will be used to type any phone number. The image below show how the view is:

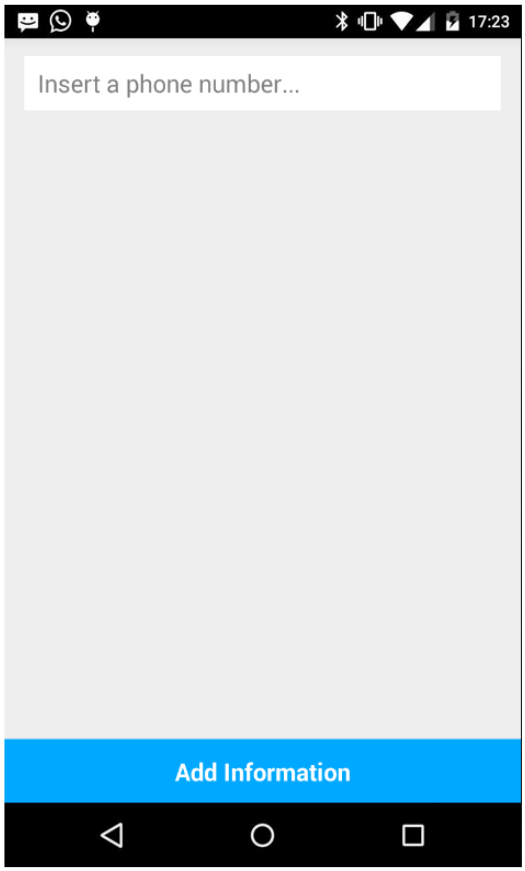


Figure 48: Settings screen 2nd prototype

This view it was made with a new Java class, which only show the components on the screen and save the phone number the user types. Apart from that and as always, this view has an XML file which has the properties of everything the view has, like the text field to type in or the button there. The code showed below explains how it was possible to make this:

```
public void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);
    setContentView(R.layout.settings_screen);
    phoneNumber = (EditText) findViewById(R.id.text_phone);

    addButton = (Button) findViewById(R.id.add_button);
    addButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            phone = phoneNumber.getText().toString();
            Intent i = new Intent(SettingsActivity.this, MainActivity.class);
            startActivity(i);
        }
    });
}
```

Figure 49: Settings screen onCreate() method

As seen on the code, the "Add Information" button stores the phone typed and take the user again to the main view. The "**setOnClickListener**" method is used to make the button works. In this case, and described before, this button makes the phone number typed stored and comes back to the main screen.

5.3.3. Final Prototype

In the final prototype one of the things changed was the settings screen. As it is possible to see in the last prototype, the settings screen only asks for a number phone and the user can store the phone typed clicking on the button there. As this project is oriented for hospitals, that means they are going to take care of many patients. For that reason, the app now asks for the name and the blood type of the user to make whoever administrate this an easy task.

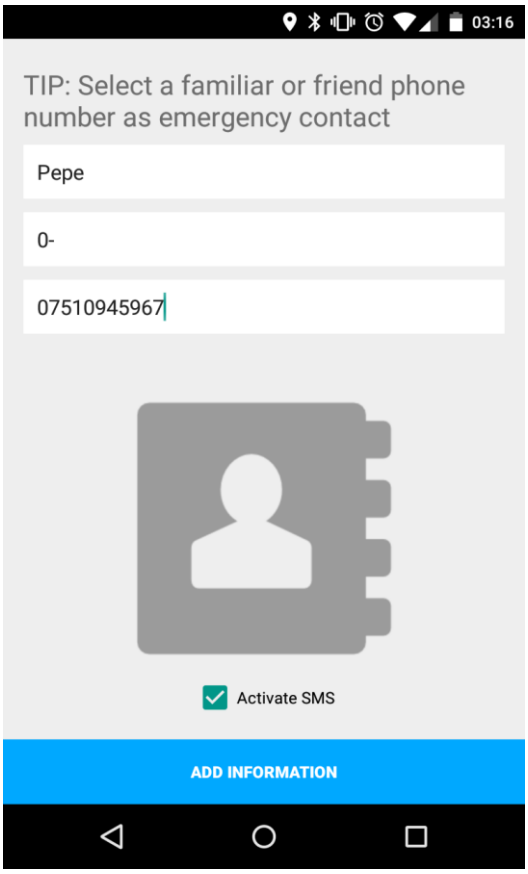


Figure 50: Settings screen Final Prototype

```
checkSMS.setOnClickListener((v) -> {

        if(checkSMS.isChecked()){

            flagSMS = true;
        }
});

if(flagSMS == false) checkSMS.setChecked(false);
```

Figure 51: Settings screen SMS flag

The code above is referring the "**Activate SMS**" checkbox. The variable **"checkSMS**" is the checkbox linked from the XML code to the Java code. Every moment the user clicks on it, will check and uncheck this box, but the main action is when this checkbox is checked, save in another variable the Boolean value "**true**", this variable is called a flag. This flag is used just to know when the app must send the SMS information. Once the app send one SMS, this checkbox will be unchecked and the user should activate again manually.

One of the new features added in the settings screen is the checkbox button to activate and deactivate the automatic SMS sending. When the app check that the BPM is above the limit wrote in the code, the app started to send automatically SMS without stop, until the BPM was normal. So this was solved with a Boolean variable which is

activated when the first SMS is sent, and deactivates the SMSs sending. Later, the user needs to go to the Settings screen in order to activate again the SMSs sending.

As seen in the image above, the huge agenda icon opens the user's agenda in the mobile phone. This makes the user an easier way to type or choose the mobile phone they want to notice in emergency case. If the user clicks on this button, every contact the user has in the phone's agenda will appear with every phone number saved on it.

It has been made to show only the contacts phone number, because the other data is not really interesting and useful for this app. Once the user has chosen any phone number, the app automatically will put this number in the phone number field. The user still needs to fill the username and blood type fields.

```
    agenda = (ImageView) findViewById(R.id.agendaButton);

    agenda.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            Intent contactPickerIntent = new Intent(Intent.ACTION_PICK, ContactsContract.Contacts.CONTENT_URI);
            contactPickerIntent.setType(ContactsContract.CommonDataKinds.Phone.CONTENT_TYPE);
            startActivityForResult(contactPickerIntent, CONTACT_PICKER_RESULT);
        }
    });
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data){

    //Link the activity started with this method
    if(requestCode == CONTACT_PICKER_RESULT){
        //If the activity was started correctly
        if(resultCode == RESULT_OK){

            //Get every contact identification and their data
            Uri getContactUri = data.getData();

            //Only shows phone number contacts
            String [] contact = {ContactsContract.CommonDataKinds.Phone.NUMBER};

            //Cursor to get the phone number chose
            Cursor cursor = getContentResolver().query(getContactUri, contact, null, null, null);
            cursor.moveToFirst();

            //Get the phone number chose
            int column = cursor.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER);
            String number = cursor.getString(column);

            //Set in the phone number field, the phone number saved
            phoneNumber.setText(number);
        }
    }
}
```

Figure 52: Settings screen agenda picker

The Java code above makes possible everything mentioned before. As we can see, the variable **agenda** is the agenda image in the Settings screen. The method "**setOnClickListener**" is necessary to define what the button is going to do, in this case, is going to open the user's agenda. As seen before, the **startActivity** method is necessary for going through screens, but in this case the app will expect some result from the agenda, so the method "**startActivityForResult**" is necessary to be used. In order to use this method, a variable has been used to give to this method an identification, and link it with the "**onActivityResult**" method. This method gets the contact identification selected and its data. In this case, every contact number phone is

shown on the list, when the user choose one of them, this method will get the phone number chosen and then, write it in the phone number field.

Finally, the last feature added in this prototype was the current user location in the SMS sent by the phone in emergency case. In the information SMS it is possible to see now the patient name, blood type, pulse rate and current location. This feature is totally essential because if the user is not in its home address, it will be easy to locate it.

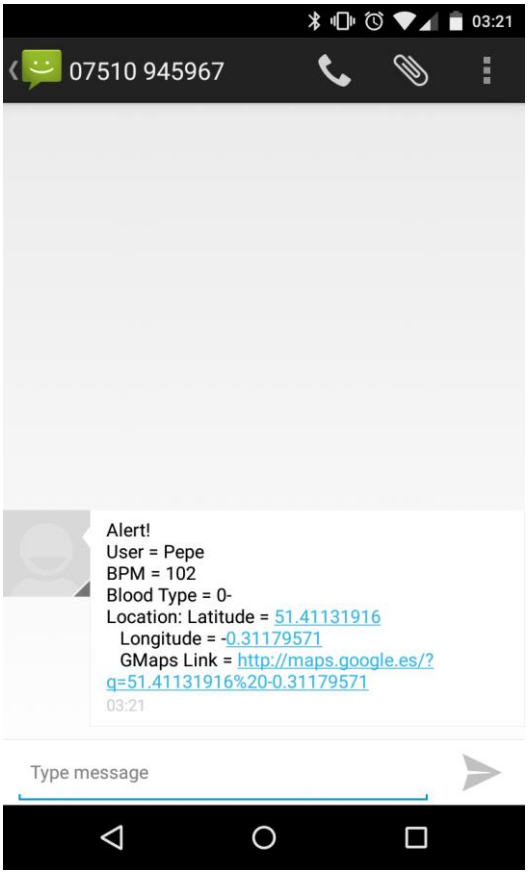


Figure 53: SMS information sent

Apart from the current location, a google maps link is provided in order to make easier to find the wristband user to the SMS receptor. The link is opened by any web browser installed in the phone and it will ask to open it with Google Maps, so the GPS can be activated immediately and locate the wristband user with it.

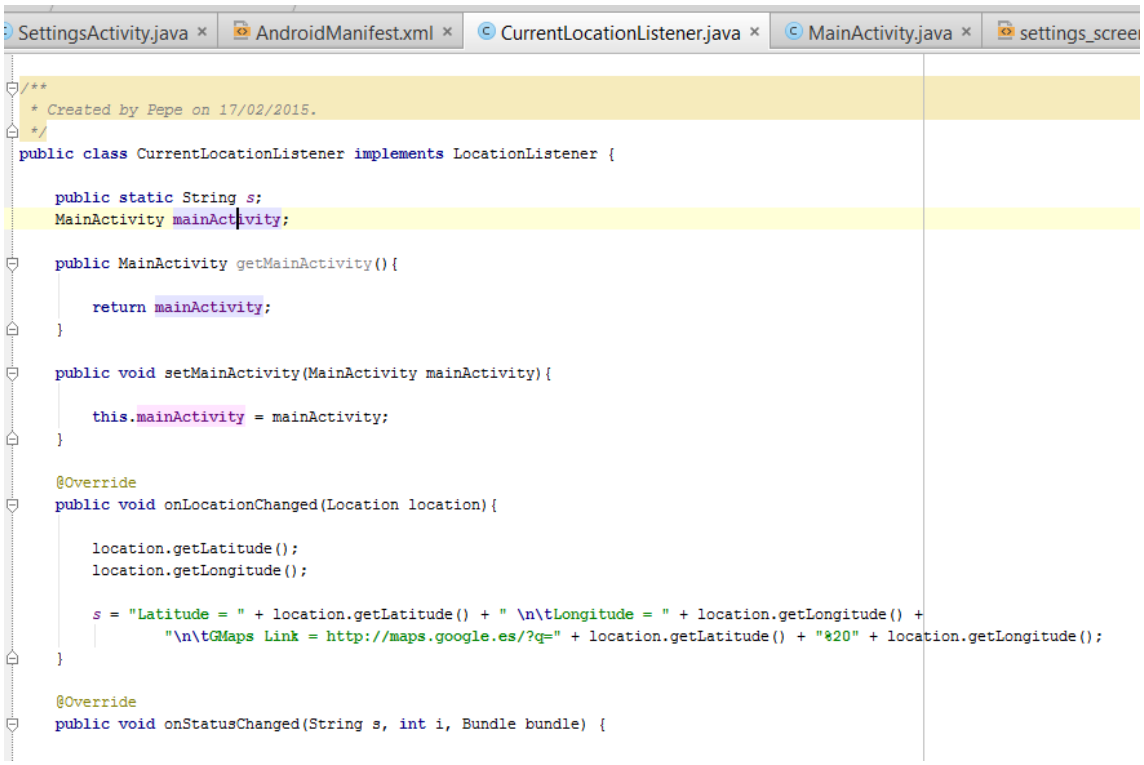


Figure 54: GPS Java class

The code above shows how to get the location has been made. In this case, it was necessary to create a new Java class and implement the "**LocationListener**" class. Once the class is created, some methods must be inside the class because of the "**LocationListener**" class, it asks for those methods, in other case it will not work. These methods are "**onLocationChanged**", "**onStatusChanged**", "**onProviderEnabled**" and "**onProviderDisabled**", but the most important in this case is the one is called "**onLocationChanged**", because when this method is called from another class, activate the GPS in the mobile phone and makes whatever is defined inside the method. In this case we obtain the coordinates and build a string to show it in the SMS.

5.4. 3D Design

As mentioned before, the 3D wristband design was the most difficult task in this project. This project is oriented to a computer science project, but it was attractive to include this new technology in order to show what it is possible to do with a 3D printer and not so much knowledge about 3D design. The software used for the design was Google Sketchup 8, with an STL Extension because it is really necessary to export the 3D model as an STL file. Apart from that, when Google Sketchup exports the model as an STL file, this usually has some mistakes or parts that are wrong, so maybe the 3D printer could interpret the model wrongly and print it bad. In order to fix these problems, it was used "netfabb", some of the purposes of this software are the possibility of getting a model preview before printing it and checking it in order to find any design error. Then, if the model has any errors, this software could fix them.

5.4.1. 1st Prototype

The first prototype was made with fixed size, this was made because at the beginning it was though about printing the wristband with flexible material, so each user could fit perfectly any wristband to its wrist. The image below shows how the first prototype was:

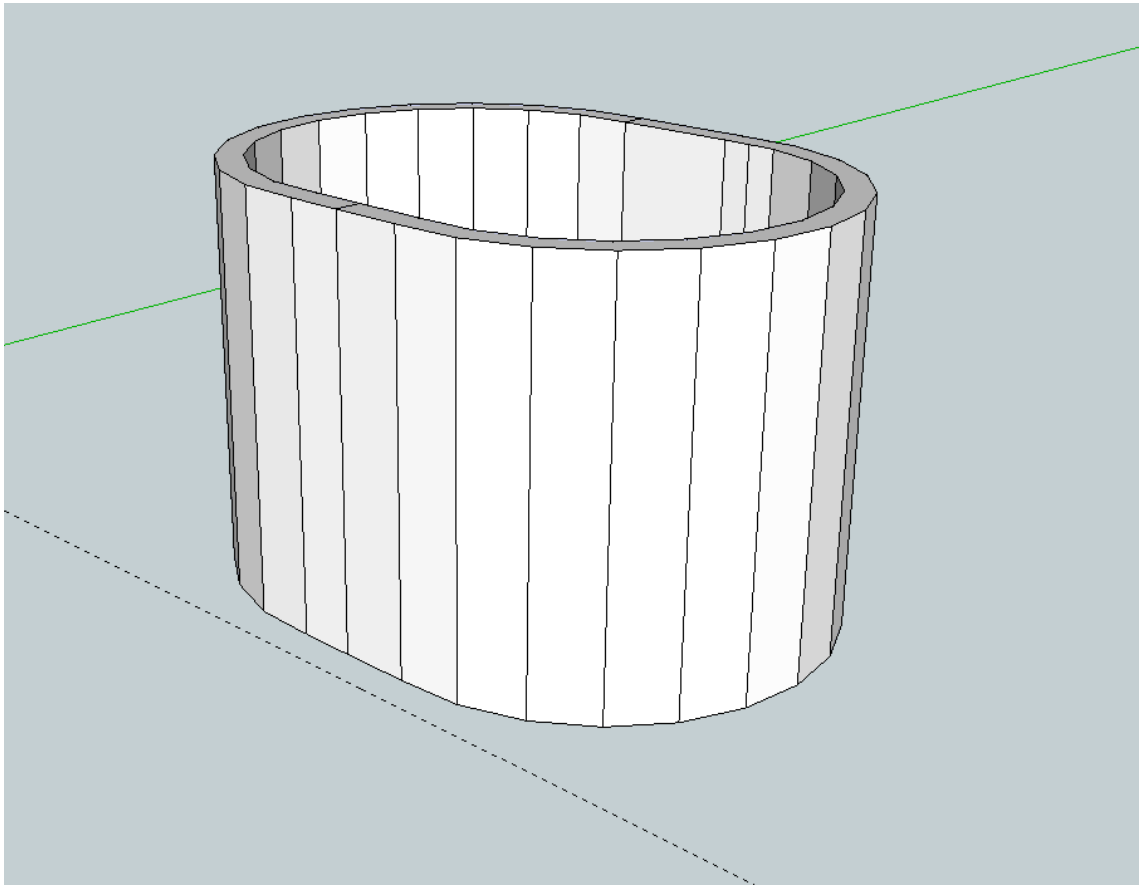


Figure 55: 3D Wristband 1st Prototype

In this case, while the wristband was being designed, the 3D printer to make print the wristband was being looked for. After a couple of emails and contact to the right person, it was known that it was impossible to print with flexible material, so the design would change in the second prototype.

5.4.2. 2nd Prototype

As mentioned before, the wristband design had to be changed to other different one because it will be printed with normal plastic. At the beginning, the first change made was to change the “close” design to an “open” one. The images below show what it means:

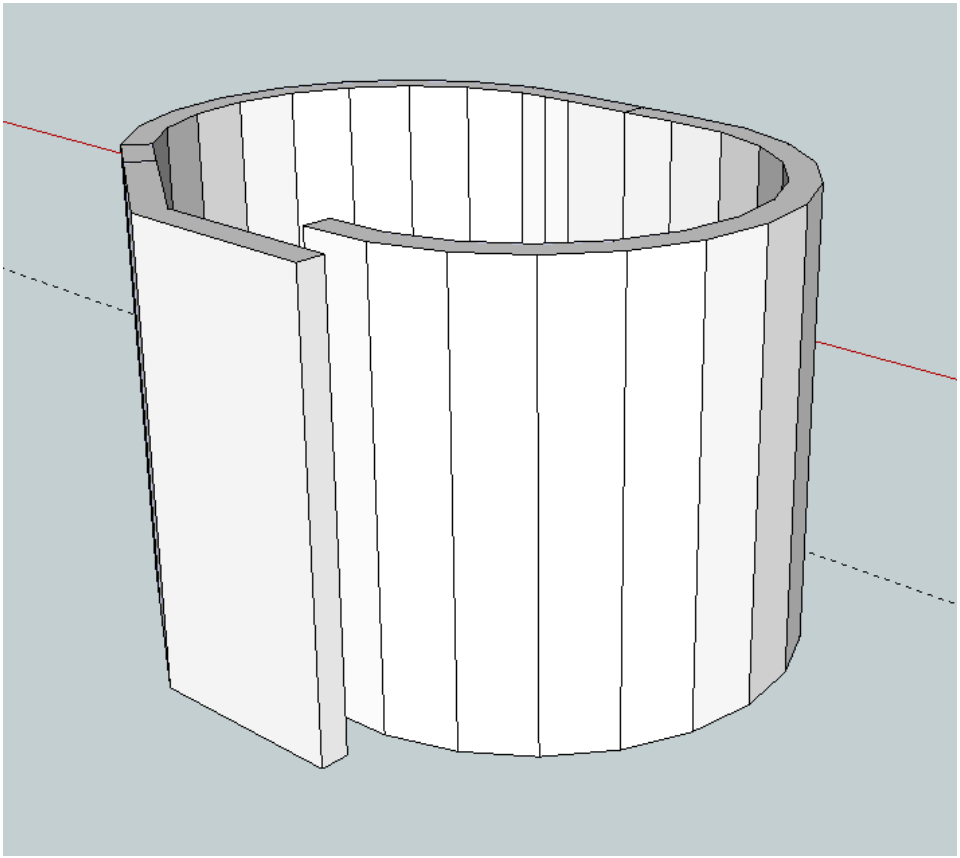


Figure 56: 3D Wristband 2nd prototype

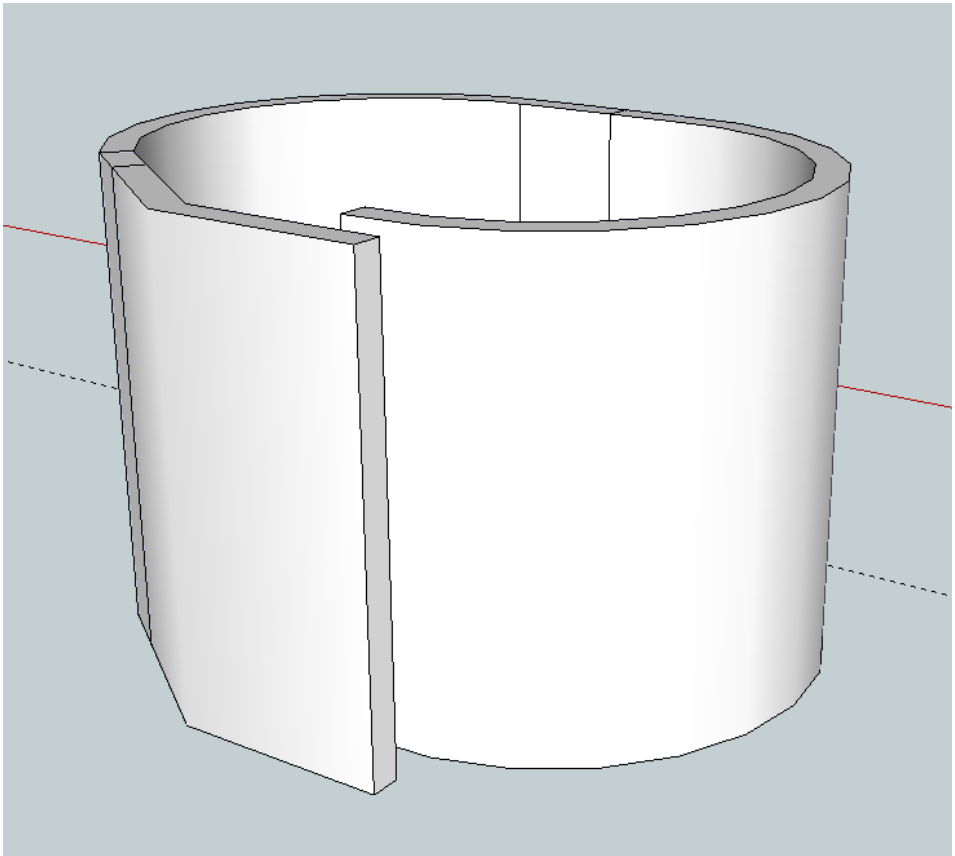


Figure 57: 3D Wristband 2nd prototype smoothed

Now, the user was able to open the wristband in order to fit into its wrist. Then, the next thing was to make some gaps and a little cylinder to allow the user to lock the wristband. The image below shows how it was made that:

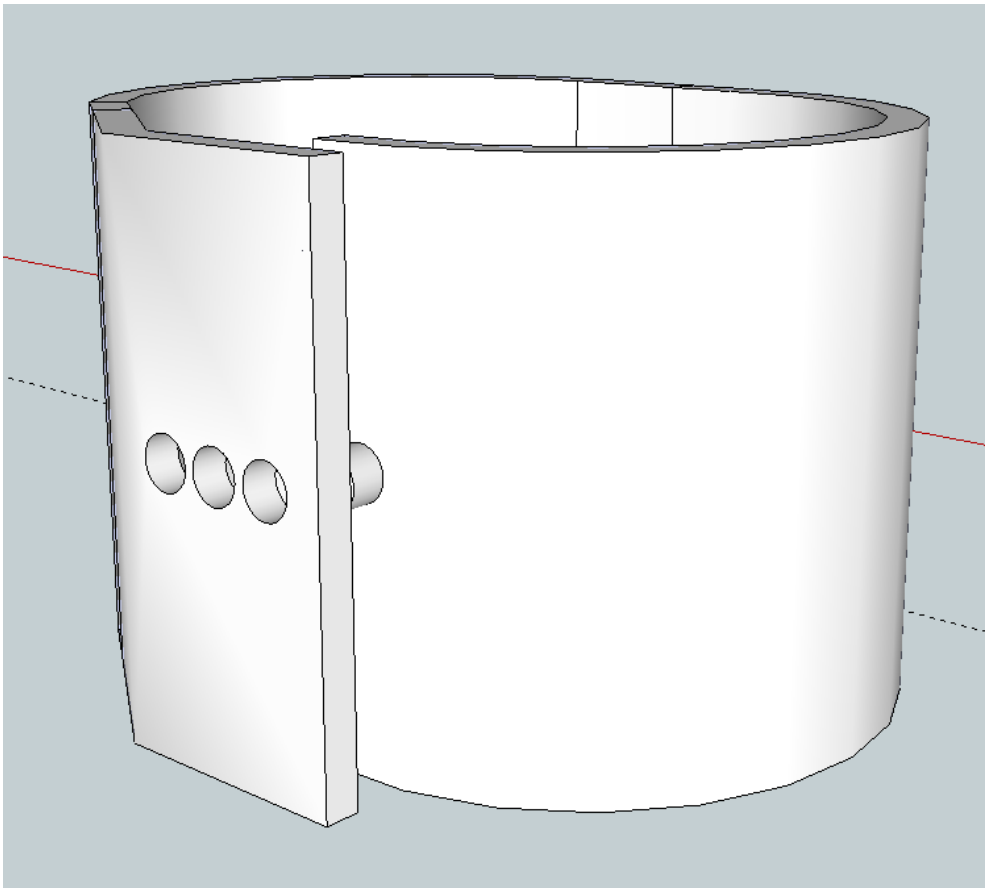


Figure 58: 3D Wristband 2nd prototype smoothed with holes

After making that modification, the user is able to lock the wristband. The 3 circles there help the user to have different sizes. This design was the first printing made to check if the design was on a good way. The results were:

Figure 59: 3D Wristband 2nd prototype printed

Figure 60: 3D Wristband 2nd prototype printed

Once the wristband was printed, it was possible to check that the size was perfect and therefore it was possible to still working on the same design. The material

used was ABS plastic, the common and cheapest material for 3D printers, but it is the most rigid material, therefore will be uncomfortable for the user in the future.

5.4.3. 3rd Prototype

As seen before, the design made on the second prototype was perfect to carry on with it. First issue fixed in this prototype was the hardness to fit the wristband on the wrist. Made with rigid material, it was necessary to change part of the design to make this task easier. As it is possible to see on the photo below, the small part with three circles has been reduced to two, as well as the size of the part itself:

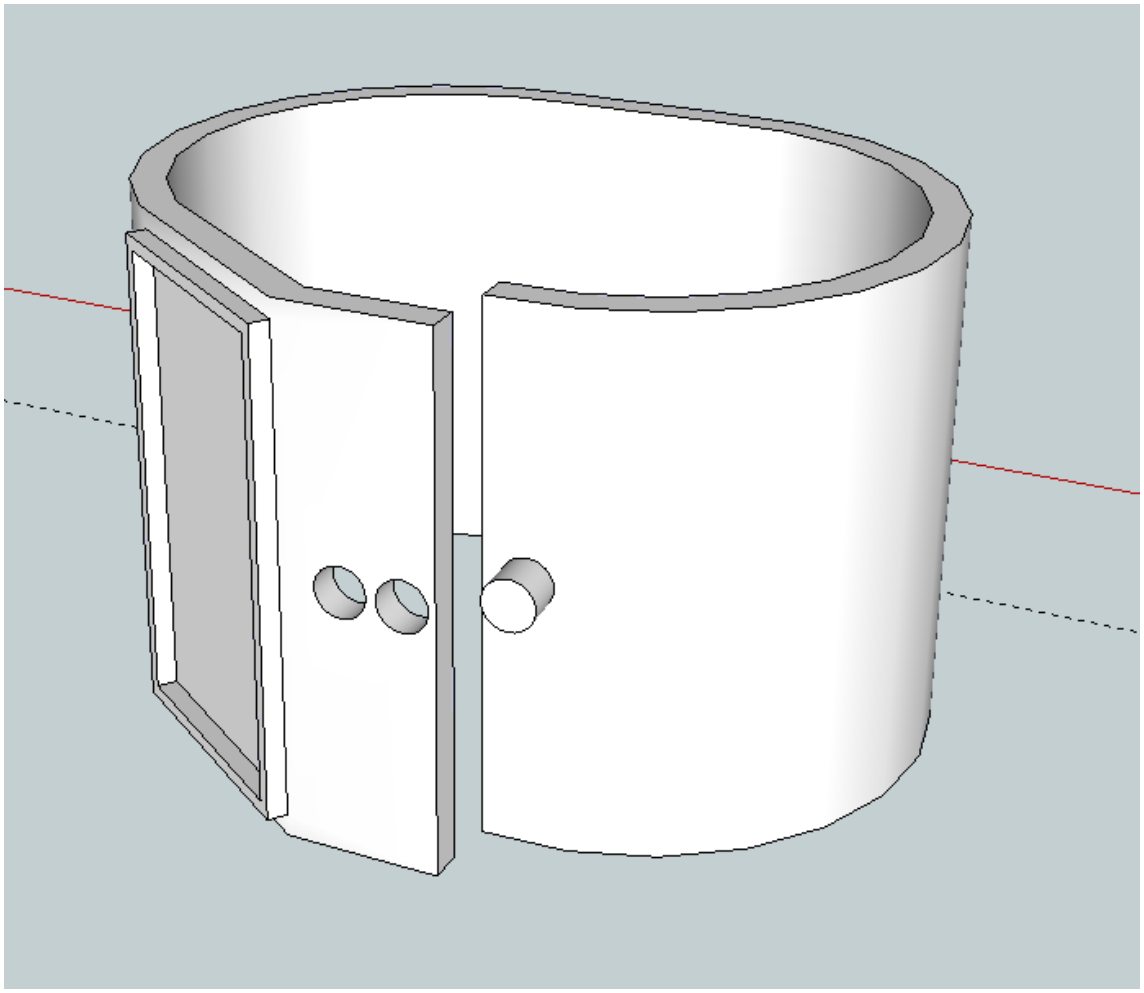


Figure 61: 3D Wristband 3rd prototype

Next to the circles is possible to see a rectangle. This space will be used to fit the Bluetooth module inside of it. It was thought to fit this module with silicone glue.

The next step was to make some slots to fit the Arduino board on the wristband. In the image below it is possible to see four cylinders emerging from the wristband. The Arduino board must be over the cylinders, and some screws were used to assemble the board. In the image below it is possible to see how the cylinders were made:

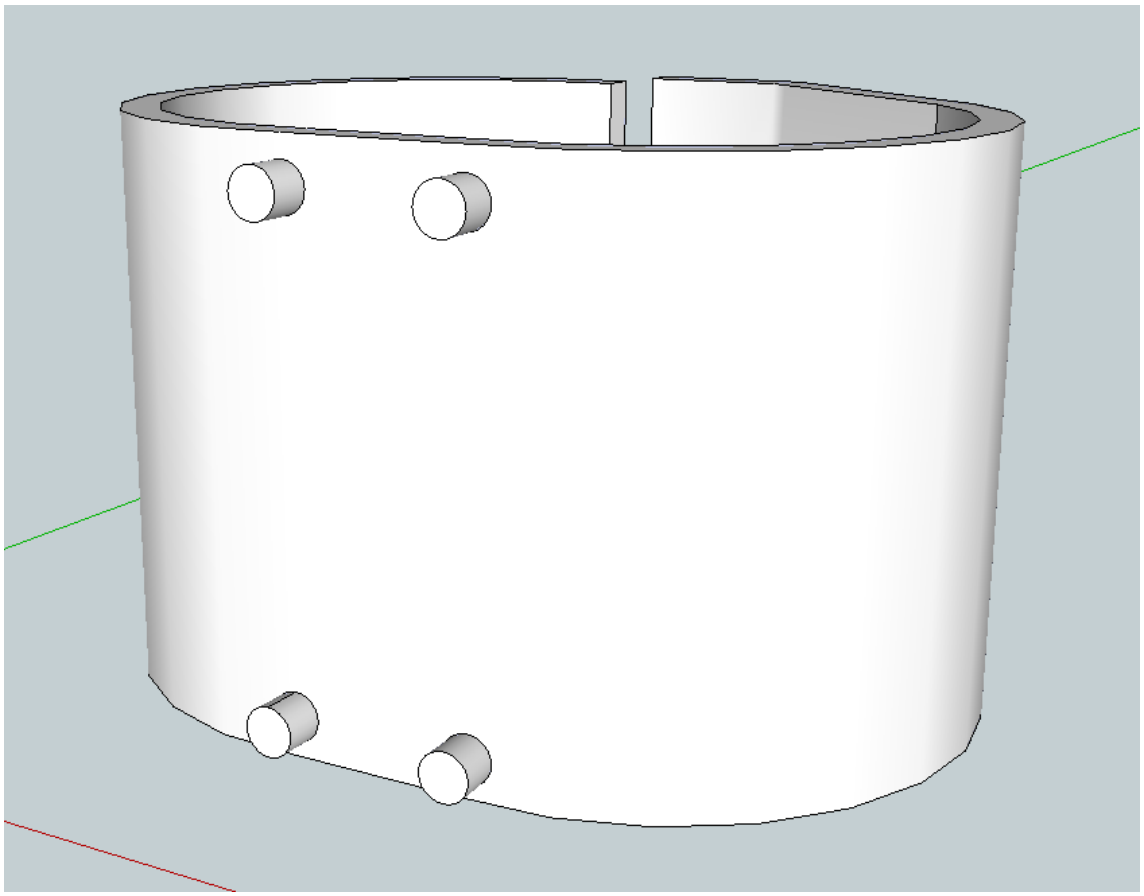


Figure 62: Arduino Nano holes

This was the most difficult task to develop in the design. Once the general design was made, Google Sketchup provides a tool to smooth curves faces. This software is not so good making new objects on smoothed surfaces, so once the general design was made and smoothed, it was difficult to make these cylinders using the design smoothed because when any of the cylinders was pulled out, made space inside of it, making an empty cylinder and the 3D printer software supposed that the cylinders were not joined with the wristband. Finally, this was solved with another tool in Google Sketchup called “Show hidden geometry”, which made easier and more accurate this task.

Finally, it was necessary to make a special cylinder inside of the wristband to assemble the sensor. This also was really difficult because it was needed to make this space as accurate as possible, because the sensor is less accurate. The sensor used here is oriented to read the pulse rate on index fingers or the ear lobes. It was necessary to bear in mind making this gap, apart from using it to assemble the sensor, using it for pressing on the wristband.

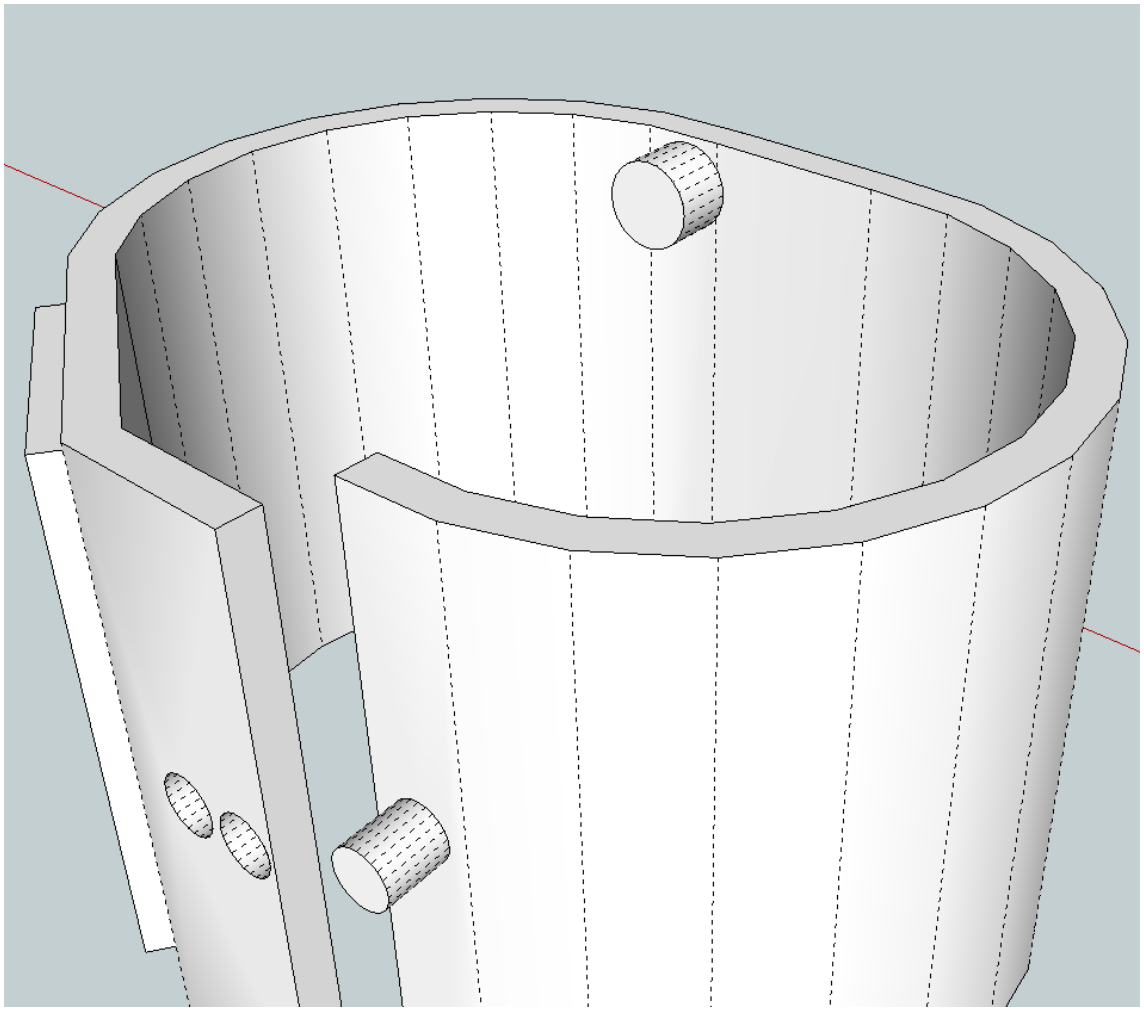


Figure 63: Sensor support

The right choice to place the sensor was the cylinder made inside the wristband because of the sensor shape and where it is possible to sense the pulse rate on the wrist.

After printing this prototype, the results were:

Figure 64: 3D Wristband 3rd prototype printed

Figure 65: 3D Wristband 3rd prototype printed

5.4.4. Final Prototype

As mentioned before, the final circuit was made with an Arduino Pro Mini board, which is much smaller than Arduino Nano board. But not also smaller, the board shape changes completely, so this forces us to change where the board is located now. It was necessary to remove all the cylinders made to fit the Arduino Nano board, and change them with a special gap for the Arduino Pro Mini board.

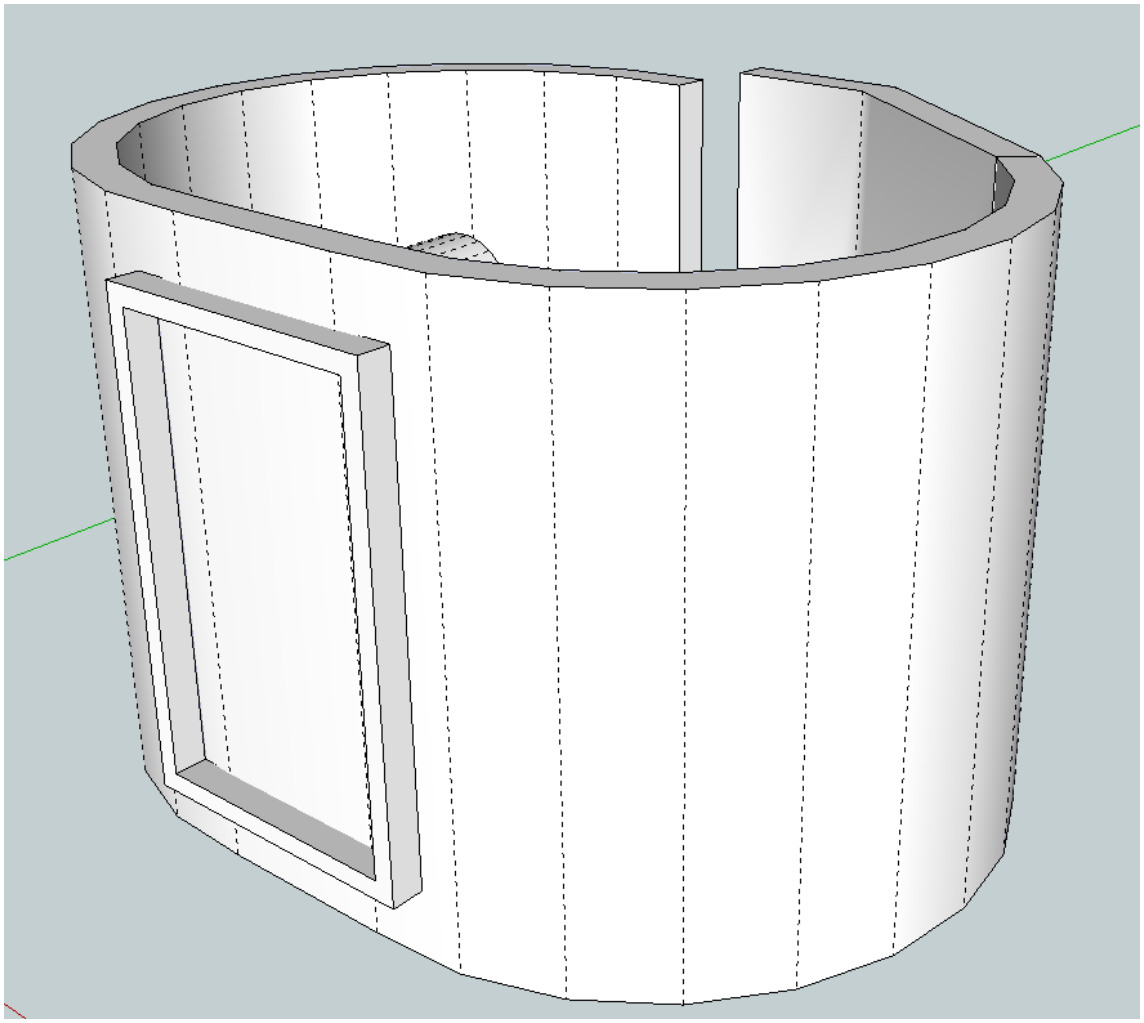


Figure 66: Arduino Pro Mini gap

As seen in the image above, the board will be in the rectangle centre. The result after printing this last prototype were:

Figure 67: 3D Wristband final prototype printed

Figure 68: 3D Wristband final prototype printed

Figure 69: 3D Wristband final prototype printed

6. Testing & Evaluation

In this section will be discussed every test made to verify that the wristband works properly and the results of them.

6.1. 1st Test

The wristband is going to be tested in this section. Every component will be tested individually and after all the entire circuit.

First of all, the main board will be tested. Just to see if it turns on and off with the batteries assembled.

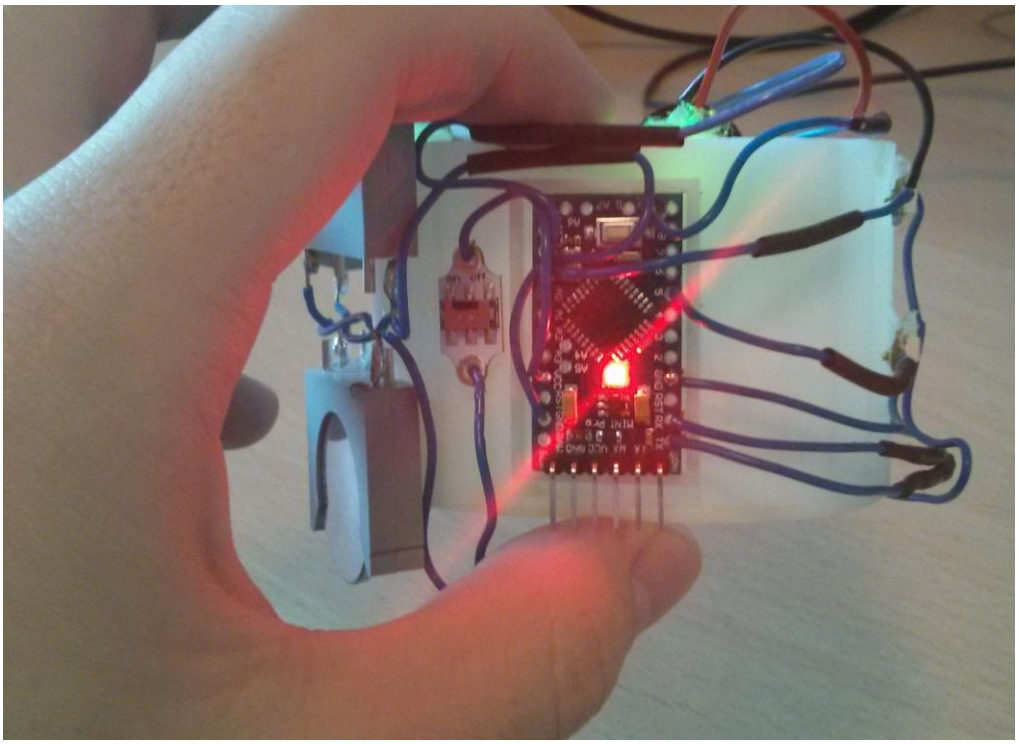


Figure 70: Switch and Arduino board turned on

As seen in the image above, using the switch included in the circuit, the board turns on and off correctly, the different LEDs on the board works properly and seems to be fine.

Secondly, the Bluetooth module needs to be checked. It is possible to know that this module works properly if the red light on it blink quickly. In the next image is possible to see that the red led blink as mentioned before:

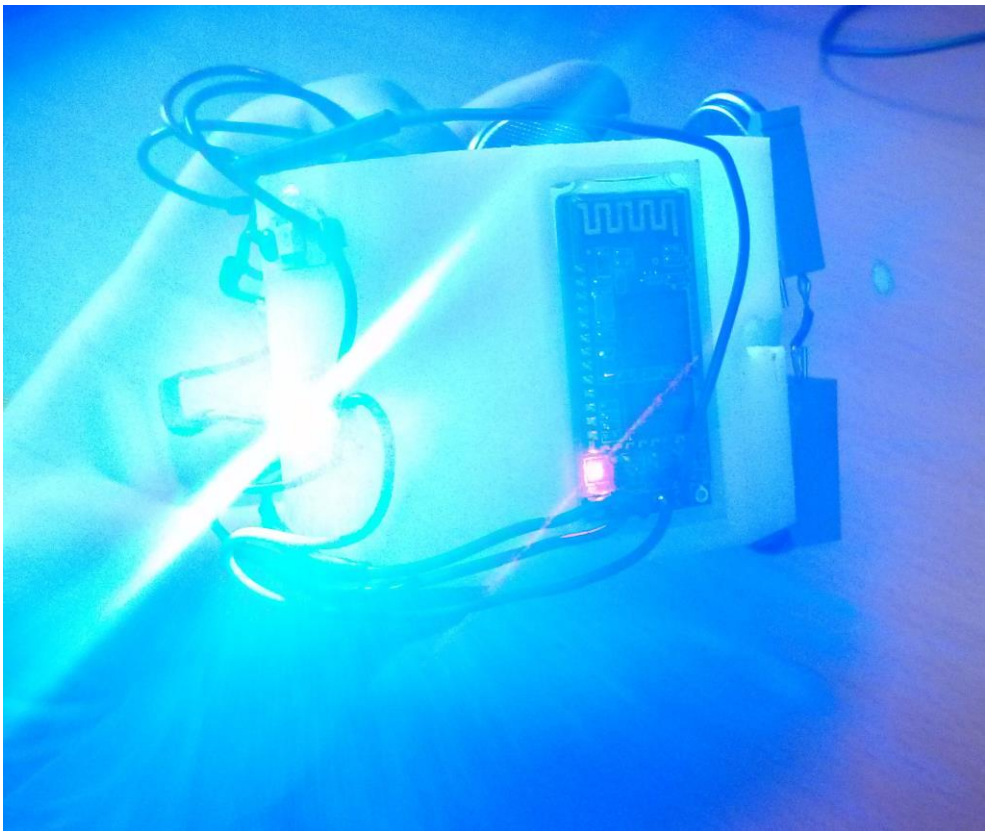


Figure 71: Bluetooth module turned on

Thirdly, both LEDs should be tested while the app is running, but for this test, they are going to be connected to the power supply directly in order to see if they turn on correctly.

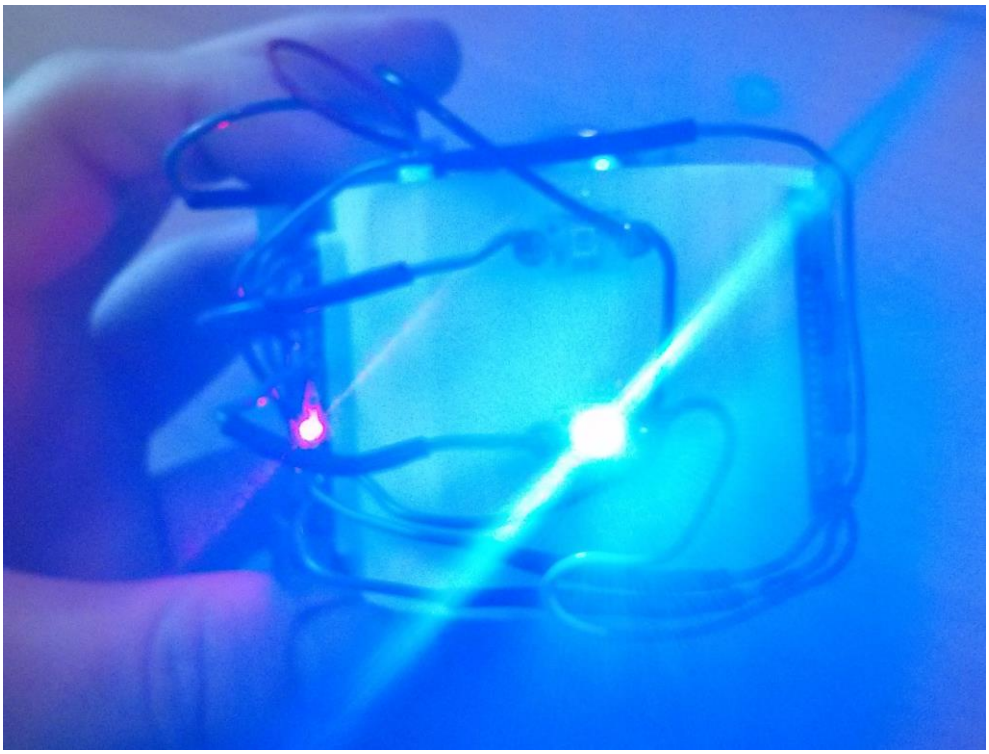


Figure 72: LEDs blinking

Finally, the green led in the centre of the sensor indicates that the sensor is working properly and is ready to use. In the image below is possible to see

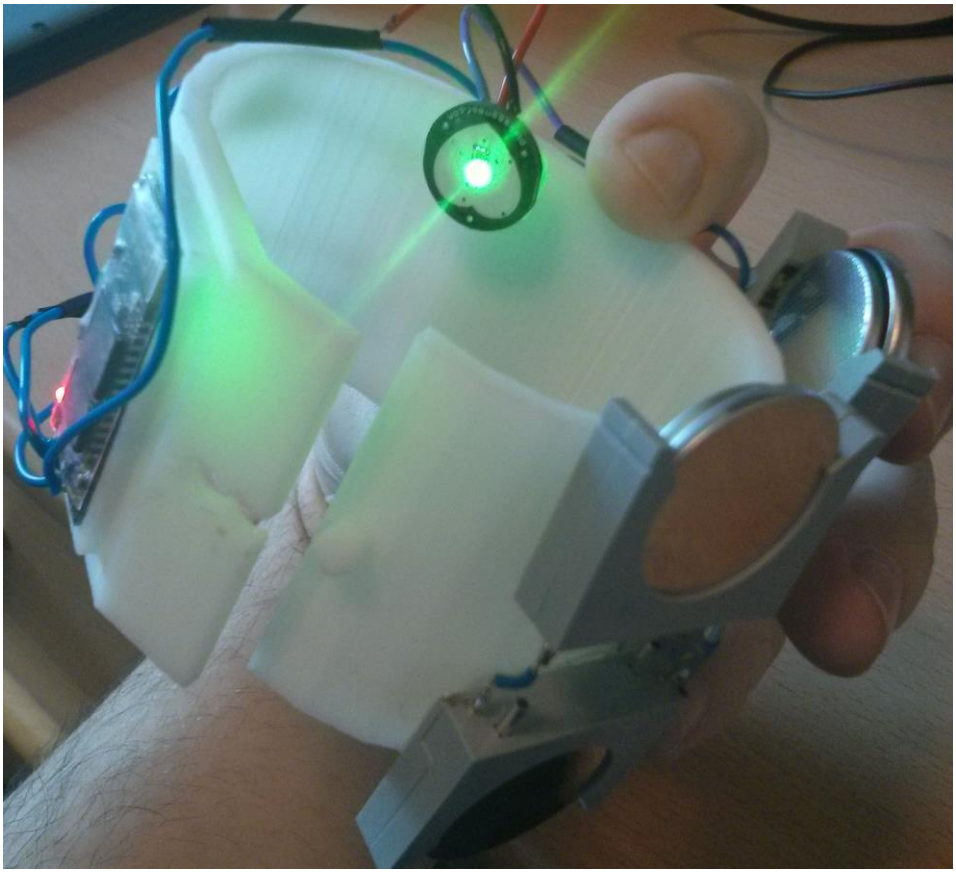


Figure 73: Pulse Sensor turned on

In addition, the batteries used finally were two coin batteries, parallel connected in order to obtain more capacity.

6.2. Final test

For this final test the entire project was tested. This means the wristband totally working with the Android app running on the mobile phone.

First of all, the wristband must be turned on before getting connected to the mobile phone. Then, before running the app, it is recommended to turn on the Bluetooth on the mobile phone and pair the wristband. Once this is done, run the app and select the wristband on the device list.

Figure 74: Device list screen final test

Once the wristband has been selected, the main screen appear showing our BPM, but the contact information in the settings screen is not configured, so the app is not going to send any SMS.

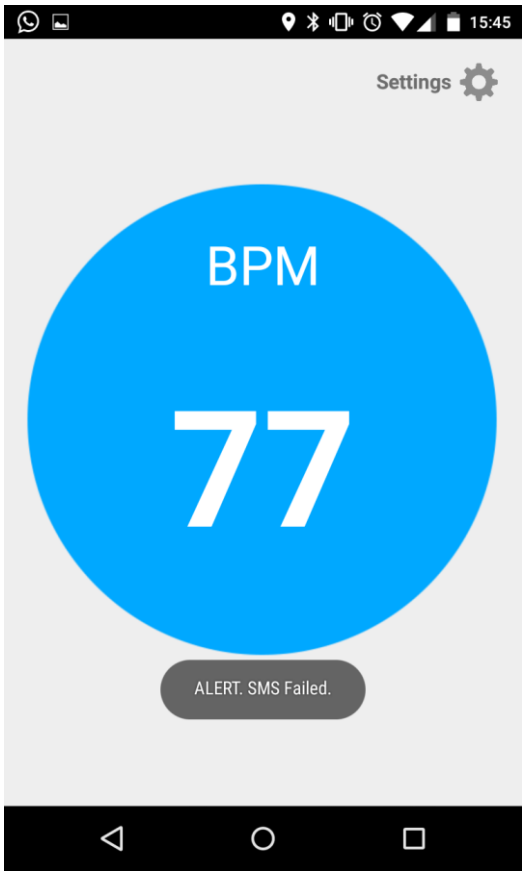


Figure 75: SMS Failed final test

Trying the Settings button in the main screen, the settings screen should appear. Now, it is possible to configure all the information about the user, and the phone number used to notice if the user has any problems.

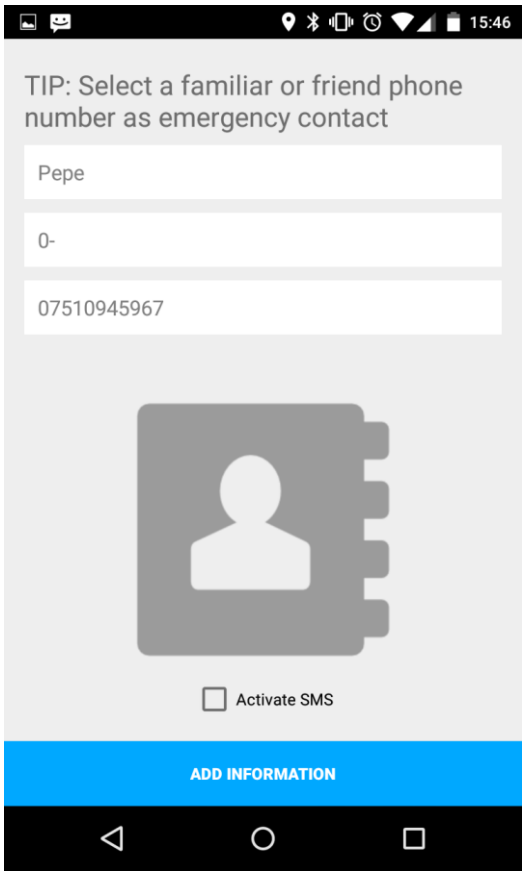


Figure 76: Information saved final test

After all the information about the user is saved and the "Add Information" button is pressed, the app comes back to the main screen. Now the app is allowed to send SMS in case the user has any problem, with the user name, blood type, BPM and the current location.

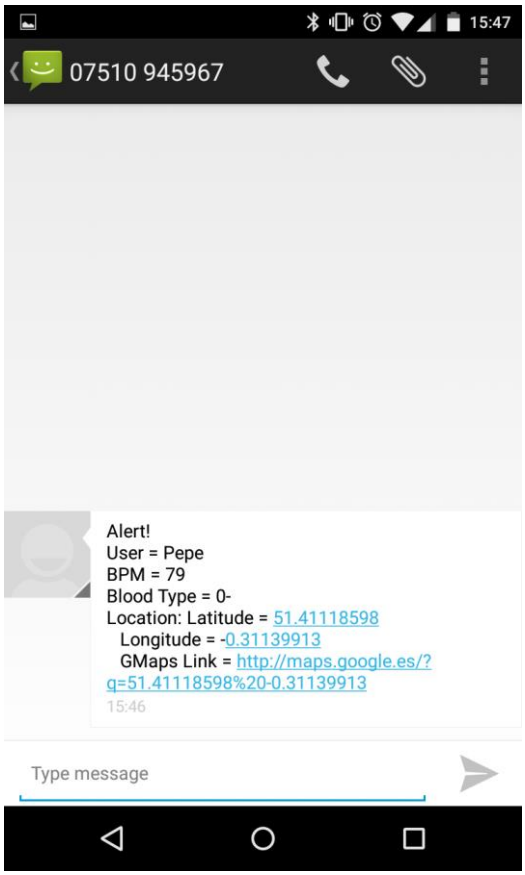


Figure 77: SMS Information sent. Final test

If the SMS sent by the app gives the correct information, the app makes its work correctly. This means that the project is completed and totally full working. Every objective has been reached with good results.

7. Critical Review & Conclusion

In this section the critical review of this project will be discussed and a conclusion about it will be given. It is going to be explained the outcome, some problems encountered developing the project, the future development of it and the production of the wristband in the case this project is totally viable to take out to the market.

7.1. Outcome

Every objective set for this project have been reached with totally good results. The wristband is totally assembled and working without any cables connected to the mobile phone, totally wireless; and the Android app is running correctly, making everything necessary for this project and for managing the data received from the wristband. However, at the beginning of this project, it had been thought to print the wristband in flexible material. Looking for it, the wristband was impossible to print with that material because Kingston University, as many other places, still does not work with flexible material already. But in fact, the plastic material the wristband has been made gave to this project good results to fit on different wrists.

Also this project taught me how really manage my time and improve some of my skills, like soldering, managing hardware and some of electronic principles. Although, it has been a really hard task developing the electronic part of this project because my electronic knowledge was limited. Apart from that, as an Erasmus exchange student, my English skills has been totally improved thanks to this project, writing this dissertation and the meetings with my supervisor. Makes me easier living here and the future on it in order to look for a job in this country.

7.2. Problems

As mentioned before, one of the main problems has been encountered was the electronic part of this project. This means how the batteries have been used, have been connected and how to manage them, apart from soldering everything. Soldering each component was not a problem, the problem was to solder them professionally. Many websites have been visited in order to learn how to connect the batteries, and what was the best way to power supply the wristband.

Secondly, the main board choice was really painful, because while this project was being developed, more and more boards appeared, and smaller than the one was choice firstly for this project. So finally, and with the possibility of changing the technology mentioned in the proposal, the main board could be changed for a new model one, really small and with the same functions.

Thirdly, the Gantt chart proposed was impossible to follow it, as mentioned before, because once the project was started, some of the tasks took more time than it was thought at the beginning.

Afterwards, and without a good solution, the pulse rate sensor accuracy is really poor. The sensor needs to be located perfectly on the wrist, in order to read properly the pulse rate.

Finally, and mentioned before, another problem was the wristband design. The problem was, every change was made, it was impossible to check if the problem was solved or not, because to check the wristband it was necessary to print a new one and see if the problem treated was solved.

7.3. Future development

As mentioned in the proposal, this project was the beginning of a biggest project. The truth is, the first proposal made and showed to my supervisor was a wristband which could read the for vital signs, but after a talk, my supervisor convinced me that this was a degree project, not a postgraduate project, so the recommendation was to make the wristband but only reading one vital sign.

As far as we know, actually hospitals receive many calls from their patients to require their services, ant the hospitals do not have any knowledge about the status of the person who is calling, or in other cases some nurses work for patients that might not require their services. In this case, the device made in this project will try to quit these phone calls and not to make these nurses work for patients that do not need this attention, making automatic patient's monitoring and reporting any problem the patient has. To manage all these wristbands, a computer would manage all the notices from these wristbands, but actually the app does not connect to any computer or anything similar, the connection is between one wristband and one Android smartphone. It is possible in the future to create an external database, used by the app to get all the information about the wristbands connected to the app. So for example, if we have in a hospital 50 patients controlled with these wristbands, each one will connect the wristband to an Android smartphone, and every smartphone would connect and send to an external database all the information gathered from each user. So managing all these devices will be easier with the external database mentioned and being monitored.

Also, after soldering the whole circuit over the wristband, it is true that there are so many cables on it, so a flexible PCB is a must, because having that flexible PCB, there will not be any cable over the wristband, except for connecting the batteries.

Finally, an ICE (in case of emergency button) would be added to the wristband, in case the communication between the wristband and the mobile phone fails, the button notice directly to any person or any hospital.

7.4. Production

As mentioned before, one of the main objectives is to have a really cheap production. This means finding the cheapest way to get every piece of the circuit and the 3D printing. In this case, all the pieces for this project were bought in Chinese websites, where the production is really cheap and the shipping to the UK is totally free,

except the sensor, which was bought in Maplin stores, because of the quality and it was necessary to have one of these immediately, when the proposal was made.

In summary, the following table shows the prices of every component used in this project:

Component	Price	Where was bought
Arduino Pro Mini	£3	Aliexpress.com
Bluetooth Module	£2.30	Aliexpress.com
Pulse Sensor	£24.99	Maplin.co.uk
Cables	£3.99	Maplin.co.uk
LEDs	£1	Kitronik.co.uk
Switch	£1.14	Kitronik.co.uk
Batteries	£2.50	Aliexpress.com
Batteries Support	£0.75	Amazon.co.uk
3D Printing	£12	Kingston University
Heat Shrink	£2	Maplin.co.uk

Apart from that, the human resources needs to be paid as well, but in this case, the human resources were the laptop used to make this project and me. In the case we need to pay an Android app developer it will be necessary to calculate the price of making this app using the following formula:

$$Final\ price = Price\ per\ hour * Total\ hours\ worked$$

Considering a junior developer salary, approximately £25000 per year and using the formula shown before, the app development will cost £2000.

7.5. Conclusion

In conclusion, making this project here in UK made me learn many different things. At the beginning I thought that I could not do this project in English, in another different language than my native language. Little by little, making this dissertation, having different meetings with my supervisor and other things made me improve my English, one of my Erasmus scholarship objectives.

Also, this project made me realize that is impossible to learn everything you want at the university. The university makes every person independent, professors try always to teach how to learn. So this project took out my capabilities, and I could demonstrate what I could do, how I could do, what I could learn and in how much time I could do everything.

In addition, thanks to this project I have learnt a lot of things about electronics, Arduino and Android, so this might not be the only project I will make with the combination of this two technologies. If it is possible, I will keep developing this project and finish it at all, as I wanted at the beginning.

8. Bibliography

3D Printer Help. (2013). *3D Printer Help*. Retrieved February 01, 2015, from http://www.3dprinterhelp.co.uk/what-materials-do-3d-printers-use/

Android, A. (n.d.). *developer.android.com*. Retrieved February 02, 2015, from http://developer.android.com/reference/android/widget/Button.html

API, A. (n.d.). *developer.android.com*. Retrieved February 05, 2015, from http://developer.android.com/reference/android/bluetooth/BluetoothDevice.html

API, A. (n.d.). *developer.android.com*. Retrieved February 05, 2015, from http://developer.android.com/guide/topics/connectivity/bluetooth.html

API, A. (n.d.). *developer.android.com*. Retrieved February 02, 2015, from http://developer.android.com/training/basics/firstapp/starting-activity.html

Brachmann, S. (2014, November 26). *www.ipwatchdog.com*. Retrieved January 12, 2015, from http://www.ipwatchdog.com/2014/11/26/a-brief-history-of-googles-android-operating-system/id=52285/

Conder, S. (2010, August 03). *code.tutsplus.com*. Retrieved February 26, 2015, from http://code.tutsplus.com/tutorials/android-essentials-using-the-contact-picker--mobile-2017

Enkeladress. (n.d.). *www.enkeladress.com*. Retrieved February 25, 2015, from http://www.enkeladress.com/article/android_snippen_show_contact_picker

Exposito, J. (2012, December 09). *expocodetech.com*. Retrieved March 03, 2015, from http://expocodetech.com/expo-tips-como-obtener-la-ubicacion-actual-con-el-gps-de-android/

Garage, E. (2011, June). *www.engineersgarage.com*. Retrieved January 21, 2015, from http://www.engineersgarage.com/articles/what-is-android-introduction

Kushner, D. (2011, October 26). *spectrum.ieee.org*. Retrieved January 20, 2015, from http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino/0

Makezine.com. (2014, September). *Makezine.com*. Retrieved January 14, 2015, from http://makezine.com/projects/connect-an-arduino-to-a-7-bluetooth-serial-module/

Maravitsas, N. (2013, May 15). *Javacodegeeks.com*. Retrieved January 25, 2015, from http://examples.javacodegeeks.com/android/core/view/onclicklistener/android-onclicklistener-example/

Point, T. (n.d.). *www.tutorialspoint.com*. Retrieved February 16, 2015, from http://www.tutorialspoint.com/android/android_sending_sms.htm

Solar, M. (n.d.). *www.mpptsolar.com*. Retrieved February 05, 2015, from http://www.mpptsolar.com/es/baterias-serie-paralelo.html

Timothy. (2014, August). *www.arduino-hacks.com*. Retrieved January 14, 2015, from http://www.arduino-hacks.com/adding-bluetooth-capability-project-arduino-hc-06/

Valera, G. (Director). (2010). *Arduino, the documentary* [Motion Picture]. Retrieved January 2015, 15, from http://vimeo.com/18539129

World Health Organization. (n.d.). Retrieved March 15, 2015, from www.who.int: http://www.who.int/mediacentre/factsheets/fs311/en/

www.accessconnect.com. (n.d.). Retrieved February 04, 2015, from http://www.accessconnect.com/images/batteries_series_parallel.jpg

www.batag.com. (n.d.). Retrieved March 15, 2015, from http://www.batag.com/images/pic/rfidwristband/wristband_rubber.jpg