

# **DEVELOPMENT OF AN ARDUINO-BASED INSOLE SYSTEM**

Connectivity via Bluetooth with an Android app based on Ionic  
framework

LAB UNIVERSITY OF APPLIED  
SCIENCES LTD  
Faculty of Technology  
Degree Programme in Information and  
Communications Technology  
Bachelor's Thesis  
Autumn 2020  
Borja Albert Gramaje

## Abstract

Author(s) Albert Gramaje, Borja	Type of publication Bachelor's thesis	Published Spring 2020
	Number of pages 57	
Title of publication <b>Development of an Arduino-based insole sensor system.</b> Connectivity via Bluetooth with an Android app based on Ionic framework.		
Name of Degree Information and Communications Technology		
<p>Abstract</p> <p>Sensors are devices that pick up physical quantities and respond to them. Their use has grown exponentially, and for certain people they are an essential part of their daily life.</p> <p>The main purpose of the thesis was to develop a prototype for a shoe which would allow us to gather detailed information on each step taken by a person.</p> <p>The reason for the development of this product was to obtain a more accurate number about the steps a person takes, and at the same time, monitor those steps, with the objective of preventing injuries or correcting habits.</p> <p>To prepare the case, a study and analysis of the market was carried out on the different devices that resembled the objective of the thesis. Then, an analysis was made to determine which hardware was necessary to make a prototype. After that, a study on which programming style (native vs hybrid) to use was made.</p> <p>An analysis of their possible development and different problems that can lead to the product development process is discussed. A study of the several hardware components is carried out. Also, a basic multiplatform app is developed to check the BLE connectivity and receive the information provided by the hardware.</p>		
Keywords Arduino, Bluetooth, BLE, Ionic Framework, Android, FreeRTOS		

## CONTENTS

LIST OF ABBREVIATIONS AND TERMINOLOGY .....	3
1 INTRODUCTION .....	4
2 SMART INSOLE DEVICE .....	5
2.1 Forciot Ltd. ....	5
2.2 Digitsole.....	5
2.3 Arion.....	6
3 MULTITASK OPERATING SYSTEM .....	8
4 REAL-TIME SYSTEMS.....	10
4.1 Types.....	10
4.1.1 Hard real-time system.....	10
4.1.2 Soft real-time system .....	11
4.2 FreeRTOS .....	11
5 HARDWARE.....	13
5.1 Arduino .....	13
5.1.1 Arduino Lilypad.....	15
5.1.2 Arduino MKR Wi-Fi 1010 .....	16
5.1.3 Bluno Beetle BLE .....	16
5.2 Sensors .....	17
5.2.1 Force sensitive resistor.....	17
5.2.2 Accelerometer .....	18
5.2.3 Level Translator Breakout.....	19
5.3 Other materials .....	20
5.3.1 Electrical resistances.....	20
5.3.2 Battery .....	21
6 ANDROID APP .....	22
6.1 Hybrid app .....	22
6.2 Angular.io .....	23
6.3 Ionic Framework.....	23
6.4 Apache Cordova.....	25
6.5 Bluetooth Low Energy.....	25
6.5.1 Ionic BLE Library .....	28
7 CASE: SMART INSOLE APP .....	29
7.1 Smart Insole App .....	29
7.1.1 Arduino hardware .....	29
7.1.2 Arduino software.....	32

7.1.3	Android software.....	34
7.2	Prototypes .....	35
7.2.1	Android app prototype v1.....	35
7.2.2	Arduino prototype v1.....	39
7.2.3	Testing and conclusions .....	41
7.2.4	Solutions and future implementations .....	44
	CONCLUSION.....	46
	LIST OF REFERENCES.....	47
	LIST OF FIGURES .....	50
	APPENDIX 1 .....	52
	APPENDIX 2 .....	55

## LIST OF ABBREVIATIONS AND TERMINOLOGY

Android: Is a mobile operating system developed by Google, based on Linux Kernel and other open source software.

API: Application program interface. Specifies how software components should interact.

CPU: Central processing unit. It is the brain of the computer, where mostly all the processes/tasks and calculations are executed.

CPU task: A program (a sequence of instructions, written to perform a specific task on a computer) in execution.

CSS: Cascading Style Sheets.

GAP: Generic Access Profile.

GATT: Generic Attribute Profile.

HTML: Hypertext Mark-up Language.

IoT: Internet of Things.

JS: JavaScript. A programming language that allows you to perform complex activities on a website.

Protocol: A set of predefined rules for the purpose of standardizing the exchange of information.

TS: Typescript. Is a superset (a language written above another language) of JavaScript.

USB: Universal Serial Bus.

## 1 INTRODUCTION

As the years go by, technology advances, and it does not do it gradually; it does it exponentially to satisfy the needs of its consumers, us.

One of the most outstanding devices that has been implemented both for professional and domestic use are the sensors. We live surrounded by sensors, working and helping to make our life much easier, for example by turning the lights on or off with a clap, detecting our presence, temperature, air pressure, humidity, speed or even detecting how much we have left in our vehicle.

Sensors are devices that can capture physical data, quantities, or other alterations of its surroundings (MecatrónicaLATAM 2019). They are used to develop smart devices such as smart insole in shoes, or smart home devices, et cetera.

These sensors can change or simplify our daily lives in an incredible way. The examples mentioned above are just a small sample of the different types of sensors that exist.

One of the fields where the use of sensors can be highlighted is the sports and health field. In the sport field, with the use of sensors, the range of information received can be increased, allowing an improvement both in nutrition and physical activity. Also, in the health field, sensors can detect the heart rate of a patient to keep him under control.

For example, by obtaining data on how a person walks, you can see if there are any problems walking. Even by counting the number of steps, you can calculate the amount of physical movement that a person is doing during the day. Health and sports are examples of fields where sensors have provided a great technological contribution.

Most of the computer systems that make use of sensors are real-time systems. It is a type of system that must give a valid response within a range of time to an event that has been generated. This event belongs to the environment in which the real-time system is operating. Systems like robots or a weapon firing control system are examples of real-time systems. (Silberschatz, Galvin, Gagne 2008, 759.)

## 2 SMART INSOLE DEVICE

With the development and characteristics of the sensors, which have been exponentially increasing their use due to their high performance, it has been possible to create intelligent devices that help us every day in some way.

One of the smart devices that stands out is the smart insole device. This is due to its high use in the lives of elite athletes, such as runners, or professional football players, who run considerable distances.

The smart insole device is a real-time system that is waiting for certain events to happen in order to respond them. The events are subject to a specific environment, being this specific for the smart insole device. These events are the steps taken by the user into the sole.

The device detects the steps that the user takes in it, providing extra information that cannot be simply obtained with the perception of our sight. With that extra information obtained, different injuries due to bad habits when running or walking can be prevented. It can also be used as a method of rehabilitation if the user has suffered a serious injury.

This product has already been developed by different companies, focusing on different areas. As they are focused on different areas, they also focus on different clients, depending on their needs.

### 2.1 Forciot Ltd.

Forciot Ltd, is a Finnish technology company created in 2015 due to their interest in sports. That interest generated a need to develop sensing technologies to make an innovation in the sports and electronic business. (Forciot 2015a.)

The company dedicated to printing stretchable electronics has already developed a smart insole device. Their system used in the product measures dynamic force and pressure distribution and it sends the information to the connected application. This measurement system consists of two elements: a force sensor for reading data and sending the data to their app for analysing the results. (Forciot 2015b.)

### 2.2 Digitsole

Digitsole, is a technology company originated in 2012 when they first launched their project of smart wearable devices. It is a company that focuses on designing and developing real-time systems, such as sports wearable devices. The client obtains specific data while

doing exercise, so that they can observe and optimize their habits to improve. (Digitsole 2012a.)

This company developed a specific product with its own software for sports people, such as runners or cyclists. The product called “Sport Profiler”, consists of a microprocessor connected to several sensors to receive the data and show it to the Digitsole application. (Digitsole 2012b).

However, this is not the only product the company has. They have various products focused on different sectors. One of these products is more focused on the health area, and another of the products monitors that the foot area is kept warm. Information on the “Sport Profiler” product can be seen in Figure 1. (Digitsole 2012b.)



Figure 1. Digitsole product aimed at sports people such as runners or cyclists

(Digitsole 2012b)

### 2.3 Arion

Arion is a technology company that focuses in developing wearables based on movement science and technology. This technology has been tested to develop high quality wearables. Their main goal is to help people improve their capabilities and health, by showing how they move. (Arion 2017a.)

Their product, also called Arion, consists of ultra-thin insoles and lightweight foot pods. A visual of the product can be seen in Figure 2. The main difference between this product and the other companies' product is that the controller or processor is outside of the insole. (Arion 2017b.)





Figure 2. Unboxing of the Arion product design (Arion 2017b)

### 3 MULTITASK OPERATING SYSTEM

A multitasking system is a system that allows the user to perform several operations, functions, or executions simultaneously. It should be noted that a multitasking system does not imply to be a concurrent (multiprocessing) system, but a concurrent system is a multitasking system, as it can execute more than one task at a time.

A concurrent system is one that allows more than one task to be executed at the same time. A concurrent operating system is one type of a multitasking operating system as it can execute more than one task due to its architecture. (Geeks for Geeks 2020.)

The main difference between a concurrent system and a multitasking system is the number of tasks executed simultaneously. In a concurrent system, more than one task is executed simultaneously, but in a multitasking system there can be a sequential flow since the system can't afford to have more than one task running due to its architecture. (freeRTOS 2020.)

A multitasking system can follow a sequential flow due to the hardware it has. It is necessary to have more than one CPU or a multicore processor to execute more than one task at a time. As in a multitasking architecture, there can be a possibility, where only a single-core processor is available so one task can be executed at a time. (Geeks for Geeks 2020.)

If a system can only execute one task at a time, it would not be a multitasking system as it cannot support that feature. This type of system, in order to give a feeling that more than one task is being executed, uses context switching. Using context switching, emulates the functionalities and characteristics of a concurrent (multiprocessing) system. (Geeks for Geeks 2020.)

A context switch is when the CPU changes its execution from one task to another. Doing that in a really short amount of time, gives the appearance or feeling of executing all of the tasks at the same time. (freeRTOS 2020.)

As mentioned above, the amount of cores provided by the system architecture, will be the number of tasks executed at the same time that the system supports.

A clear example of concurrent task execution can be seen in Figure 3, and sequential execution with various context switches (multitasking) can be seen in Figure 4

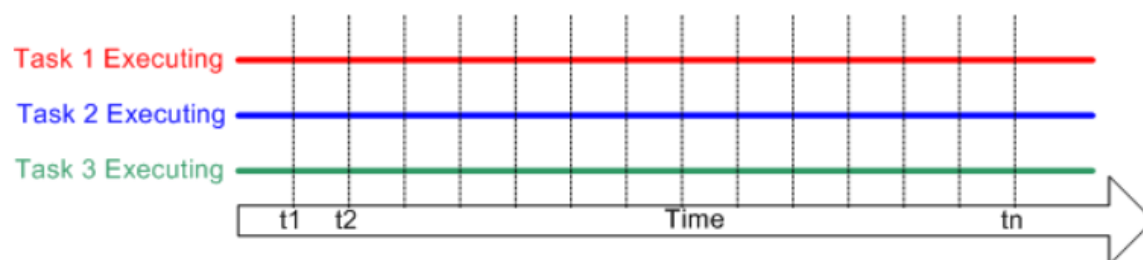


Figure 3. Example of a concurrent task execution (freeRTOS 2020)

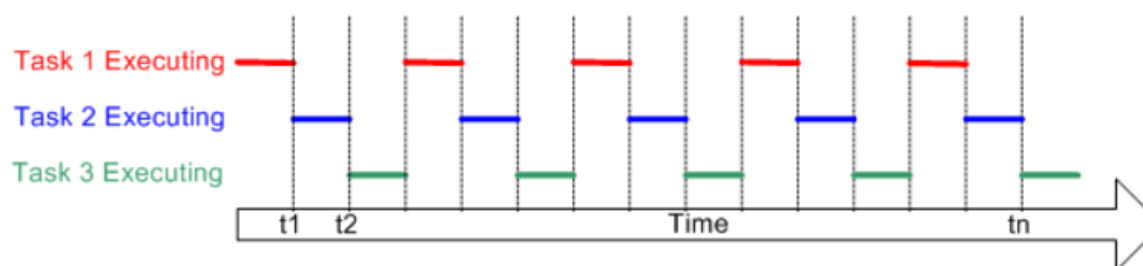


Figure 4. Example of a sequential task execution with context switching (freeRTOS 2020)

## 4 REAL-TIME SYSTEMS

In a real-time system, time is a factor to be considered, and precisely, it is not unimportant, but rather, a crucial factor. A real-time system is a computer system that apart from requiring the results or answers from an event to be correct, they must be given in a specified deadline period. (Silberschatz et al 2018, 759.)

A real-time system has been described as one that understands the environment by receiving, processing, and returning the data as results.

Although the system provides us with correct answers or results to a certain event, if that answer arrives later than the minimum time marked as deadline, despite being correct, will not be treated as a valid real answer. (Silberschatz et al 2018, 759.)

These systems are also identified as safety-critical systems. This is due to the types of answers that the system gives to certain events, being these critical events that need an immediate critical response. (Silberschatz et al 2018, 760.)

In these specific real-time systems that interact with a critical environment, they must respond to the event correctly and quickly before time runs out. If this event is not responded correctly and quickly enough, a catastrophe could occur. Some examples are any weapon system used by the army or a flight controller, because if the response is late or not correct, as mention before, a catastrophe could occur. (Silberschatz et al 2018, 760.)

Real-time systems are used in an environment with many events that needs to be accepted and process to make a solution in a very short time (depending the restriction). An example, for a smart insole device, its environment would be when the user makes any kind of movement, either walking or running. The events are the forces exerted in the insole.

### 4.1 Types

There are two different types of real-time systems: soft and hard. These types are depending in how strongest the requirements are, as well as what happens if an answer misses a deadline.

#### 4.1.1 Hard real-time system

This system has the most stringent requirements and cannot miss the deadline of responding and giving an answer to an event originated in the environment. Missing the

deadline in this system can be disastrous, producing a huge degradation in the system and there could be a catastrophe due to passing the deadline. (Silberschatz et al 2018, 760.)

As mentioned before, there are several real-time systems that interact with stricter and critical environments. These systems need to give correct answers and without going beyond the established time limit. So, a safety-critical system, that neither can pass the time limit, is a type of a hard real-time system.

An image of a hard-real-time system as well as a safety-critical system, in this case a flight system controller, can be seen in Figure 5



Figure 5. Flight system controllers (Quora 2017)

#### 4.1.2 Soft real-time system

This system is less restrictive than the hard real-time system. It can occasionally miss the deadline of responding and giving an answer to an event originated in the environment. For these systems, missing the deadline have no disastrous consequences and no degradation is generated. An example of a soft real-time system is a video delivery software. (Silberschatz et al 2018, 760.)

#### 4.2 FreeRTOS

FreeRTOS is a free to use library compatible with Arduino that allows the programmer to program and build a real-time system. FreeRTOS is an abbreviation of "free Real-time operating system".

*An operating system is a program that manages the computer hardware. (Silberschatz et al 2018, 3).*

This library allows to create different tasks, with different priorities, to control which task it should execute first when it has the opportunity. It also includes its own task scheduler, for the multitasking programming of the system.

Each task created with the specification done by FreeRTOS, is a small program. It has an entry point, but it will not end as the task is always running infinitely. So, if a task has only one entry point and no exit point because it is in an infinite loop, there is a need in how to do a multitask programming. (Barry 2016, 46.)

The tasks using this library, can have two different states: running and not running. The first state refers to the task that is being executed. The other state refers to all the suspended or delayed tasks which are in queue waiting to be changed of state and executed. (Barry 2016, 48.)

The programmer, through certain calls to different functions explained and written in the FreeRTOS API, can change change the status of the tasks. So a task that is running can be suspended with a call, in order to make a context switch between the tasks generated in the library.

If it is the case that there are two tasks with the same timeframe/or priority to be executed, then the scheduler programmed in the library will decide which of the two tasks should proceed and change its status to execution.

## 5 HARDWARE

All smart insole devices developed by different companies, whose product is on sale, use specific hardware. This hardware is composed of several components. These components are sensors, which must detect the event and react to it, and a controller, which allows us to process the data obtained by the sensors.

One of the most worldwide used boards that include a controller to allow the processing of data received by the sensors is Arduino. Arduino enables the programming of the functionalities of a microcontroller. It works with the data from the sensors, and it manages the data in the way it is programmed.

However, the board is not the only thing needed. Various sensors are also required to detect movement and pressure in the sole. This is the two key factors in detecting a step.

### 5.1 Arduino

Arduino is an electronic open-source platform used to build electronic devices or projects. Consists of both physical programmable circuit board and a software to program it. The circuit board is also known as a microcontroller. It is a circuit board where sensors or other accessories such as ethernet modules or Bluetooth modules can be connected, et cetera. An image of the microcontroller can be seen if Figure 6. (Arduino 2005a.)



Figure 6. Arduino UNO Rev3 microcontroller (Arduino 2005c)

To program the board and all the accessories such as sensors or modules, the software is needed. This piece of software is called Arduino IDE. Arduino IDE is an open-source Arduino software programmed in Java, that facilitates the work to write code and upload that code to the physical Arduino board. (Arduino 2005b.)

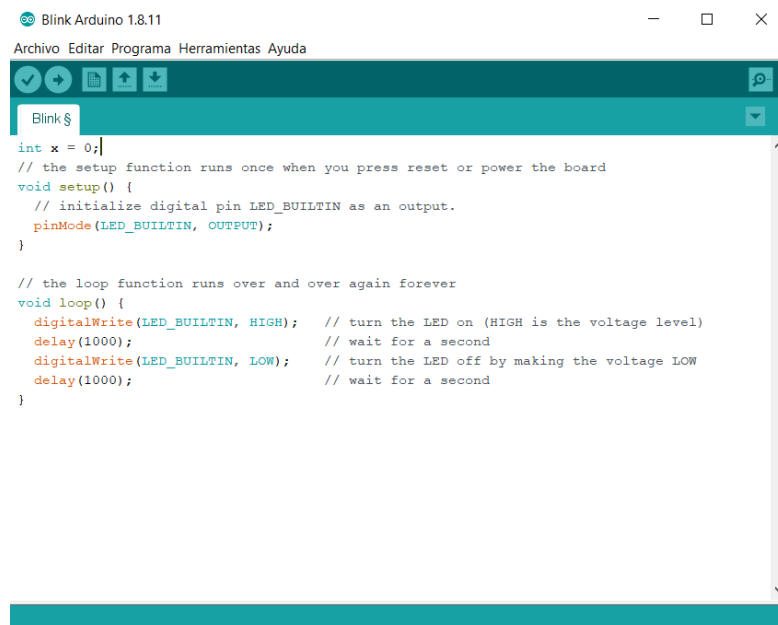
Arduino IDE programming language is in C/C++, and it includes plenty of libraries that facilitate the work of writing code to program the circuit board. The interface of the program can be seen in Figure 7. (Arduino 2005b.)

Arduino programs always follow the same path. The code is always divided into three parts:

- Library and variable declaration
- Setup()
- Loop()

Library and variable declaration are the sections where variables are declared and assigned. In this section is where libraries are added to use them for specific sensors or modules that are plugged in into the microcontroller. (Arduino 2005b.)

The Setup() function, runs once all the commands inside of the function when you press reset or you power the board. The loop() function, is a function that runs over and over all the commands inside of it infinitely. (Arduino 2005b.)



The screenshot shows the Arduino IDE interface with the title bar 'Blink Arduino 1.8.11'. The menu bar includes 'Archivo', 'Editar', 'Programa', 'Herramientas', and 'Ayuda'. The toolbar contains icons for opening, saving, and running. The code editor displays the following C++ code for the Blink example:

```
int x = 0;
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
  digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay(1000); // wait for a second
  digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
  delay(1000); // wait for a second
}
```

Figure 7. Arduino IDE interface and blink example already done by Arduino (Arduino 2005b)

Different analysis has been made between different Arduino boards, among them are the Bluno Beetle BLE, Arduino Lilypad and the Arduino MKR Wi-Fi 1010.



### 5.1.1 Arduino Lilypad

The Arduino Lilypad is designed for wearables and textiles projects. But the problem, according to Arduino's official website, is that it is a retired product, which cannot be purchased from the official website. It is possible to buy it from other suppliers. An image of the Arduino Lilypad can be seen in Figure 8. (Arduino 2005d.)

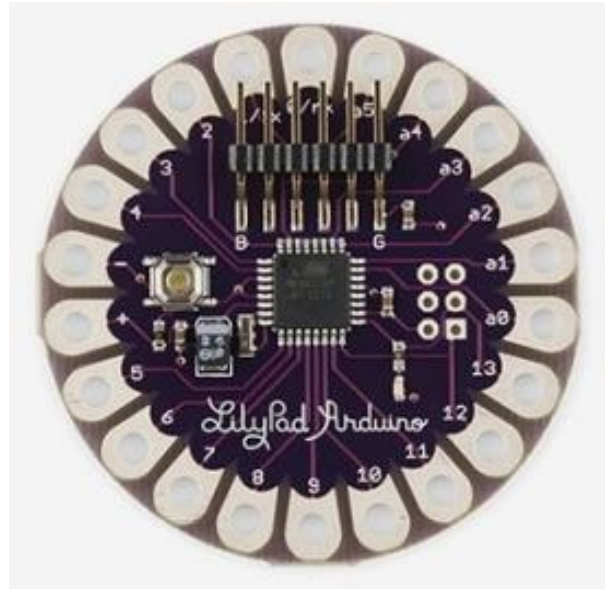


Figure 8. Official sketch of the Arduino Lilypad (Arduino 2005d)

The board was designed and developed by Leah Buechley and SparkFun Electronics, and it is based on the ATmega168V microcontroller. All the technical specs of the board can be seen in Figure 9. (Arduino 2005d.)

Microcontroller	ATmega168 or ATmega328V
Operating Voltage	2.7-5.5 V
Input Voltage	2.7-5.5 V
Digital I/O Pins	14
PWM Channels	6
Analog Input Channels	6
DC Current per I/O Pin	40 mA
Flash Memory	16 KB (of which 2 KB used by bootloader)
SRAM	1 KB
EEPROM	512 bytes
Clock Speed	8 MHz

Figure 9. Technical specs of the Arduino Lilypad (Arduino 2005d)

### 5.1.2 Arduino MKR Wi-Fi 1010

The Arduino MKR Wi-Fi 1010 is an official Arduino product, and it is not retired from the official website, as it was the Arduino Lilypad. Likewise, Arduino's website recommends the use of this device if the goal is to send data via bluetooth to our mobile device, or even to set up a small IoT (Internet of Things) network of applications. (Arduino 2005e.)

This device offers 3 ways of wireless communication, which are: one way through Wi-Fi, and two ways through Bluetooth, which are the classic Bluetooth, and the Low Energy Bluetooth. An image of the Arduino MKR Wi-Fi 1010 can be seen in Figure 10. (Arduino 2005e.)

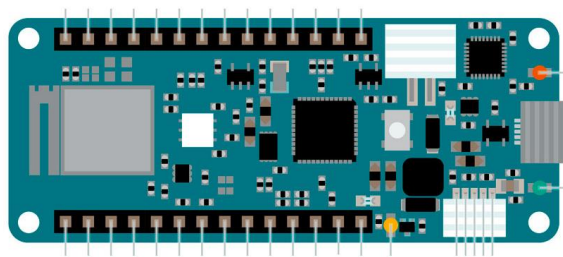


Figure 10. Arduino MKR Wi-Fi 1010 (Arduino 2005e)

What makes this product attractive, is the intrinsic incorporation to the board of different wireless connection possibilities. Avoiding the need in buying a wireless connection module and its size. The size is slightly larger than the Lilypad, in turn, being its size larger, allows more ports or connection pins on the board.

### 5.1.3 Bluno Beetle BLE

This board is developed and sold by DFROBOT and is based on Arduino. The programming that is done to an Arduino is done in the same way to the Bluno Beetle BLE. This board is like Arduino lilypad, but with the difference that already includes a bluetooth module for wireless connectivity. A sketch of the board can be seen in Figure 11. (DFROBOT 2020.)

The board is built with the microcontroller ATmega328P, that is one of the most common microcontrollers that the official Arduino boards have. This board also has different types of pins to connect different devices, shields, accessories, or wires to it. (DFROBOT 2020.)

It also has the CC2540 from Texas Instruments which functions as the core of the bluetooth wireless interface. The specifications on the board can be seen in Figure 12. (DFROBOT 2020.)

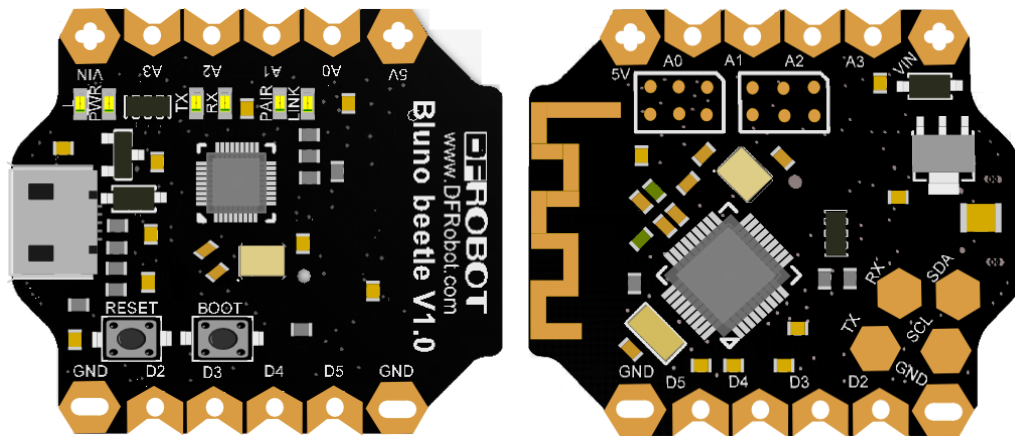


Figure 11. Bluno Beetle BLE (DFROBOT 2020)

Bluetooth Chip	CC2540	Digital Pin	x4
Sensitivity	(-93dBm)	Analog Pin	x4
Working Temperature:	(-10 °C ~ +85 °C )	PWM Output	x2
Maximun Distance	50m(Open field)	UART interface	x1
Microcontroller:	ATmega328P	I2C interface	x1
Clock frequency:	16 MHz	Micro USB interface	x1
Working voltage:	5V DC	Power port	x2

Figure 12. Bluno Beetle BLE specs and connectivity settings (DFROBOT 2020)

## 5.2 Sensors

Sensor is a device that can detect external actions or stimulus and respond to them accordingly. It gives us the ability to capture information from the physical environment around us. (MecatrónicaLATAM 2019.)

They oversee transforming the physical magnitudes of the environment into electrical signals that are interpreted by a microcontroller embedded in a circuit board (Guimerans 2020).

### 5.2.1 Force sensitive resistor

These are the simple sensors, as they are "switches" that are activated or deactivated if they encounter an object or a pressure is applied. These resistors allow us to detect physical pressure, weight or squeezing.

They are not accurate, they give a possible range of responses, but there is not a precise answer. These resistors are not a good option if the main objective is to measure the exact weight of an object or person. (Perabo 2016.)

To detect when a pressure has been applied to the resistor, a change in its resistance will produce, due to the pressure exerted. The resistance changing variation will be inversely proportional to the applied force (Perabo 2016).

A structure of how the force sensitive resistors are, can be seen in Figure 13

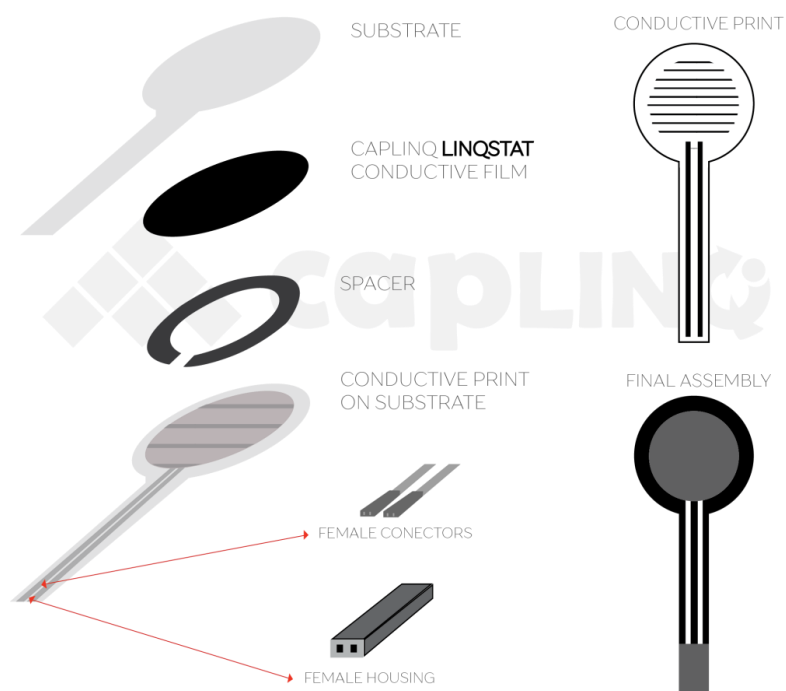


Figure 13. Building structure of a force sensitive resistor (Perabo 2016)

When a force is applied to the sensor, the conductive film is deforming with the substrate. The air generated passes through the spacer, and the conductive film contacts the conductive print on the substrate. The more it gets touch; the lower value of the resistance will be. (Perabo 2016.)

### 5.2.2 Accelerometer

They are devices that measure acceleration, which is the change in speed over a period of an object. These sensors sense static or dynamic acceleration forces. Static forces include gravity, while dynamic forces include vibration or motion. (Adafruit 2013a.)

Accelerometers can measure acceleration in one (X-axis), two (X- and Y-axis), or three (X-, Y-, and Z-axis). An image of the accelerator can be seen in Figure 14. (Paguayo 2019.)

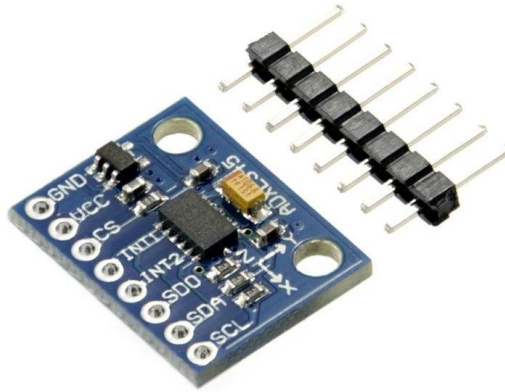


Figure 14. ADXL345 Digital accelerometer (Adafruit 2013a)

### 5.2.3 Level Translator Breakout

Level Translator Breakout is a device that normally changes the voltage of a current. It is usually used in problems of incompatibility between devices since they can operate at different voltages. If an incorrect voltage is applied to the device that operates in a lower voltage, the device may melt. (SparkFun 2020.)

The model “PCA9306”, produced and distributed by SparkFun is the one example of a Level Translator Breakout. An image of the product can be seen in Figure 15.

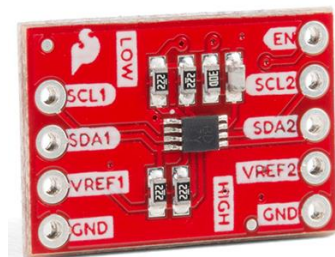


Figure 15. PCA9306 - SparkFun Level Translator Breakout (SparkFun 2020)

The board is divided into two parts, a low and a high part, the high part that tolerates up to 5V will be connected to the circuit supplier, and the other part (the low), will go to the component that does not tolerate 5V.

## 5.3 Other materials

### 5.3.1 Electrical resistances

When connecting the various sensors to the Arduino, it is necessary to use electrical resistors between components. This resistance, measured in ohms ( $\Omega$ ), is opposed to the flow of electrical current through a conduit. To know which value in ohms to use for the connection of the sensors, Ohm's law is used. (ResistorGuide 2019.)

Ohm's law states that the voltage of the circuit is equal to the current "I" in amps (A) times the resistance "R" in ohms ( $\Omega$ ):

$$V = I \times R$$

From this equation you can calculate current as  $I = \frac{V}{R}$ , and the resistance as  $R = \frac{V}{I}$ .

In the force sensitive resistors case, a voltage divider circuit is needed. It results in a variable voltage output read by the Arduino microcontroller board. The voltage divider circuit scheme can be seen in Figure 16. (Adafruit 2012b.)

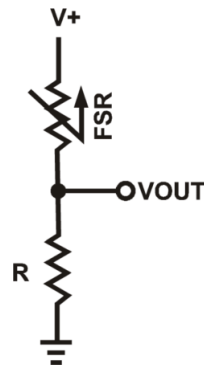


Figure 16. Voltage divider circuit scheme using force sensitive resistors (Adafruit 2012b)

The VOUT that is the resulting output, can be calculated as:

$$V_{out} = V_{cc} \times \left( \frac{R}{R + FSR} \right)$$

The measuring resistor, R, is chosen depending in how much we want to maximize the desired force sensitivity range (Adafruit 2012b).

### 5.3.2 Battery

Generally, the easiest way to power an Arduino is through the USB port, although the USB port may not be the most optimal way to power the board when delivering a final prototype.

As mentioned, the USB port is one of the most common ways to power the board, being connected via USB to our computer. In turn, not only the computer is the only way to power via USB, but also can make use of emergency batteries for mobile devices.

At the same time, the Arduino has two pins nicknamed GND and Vin. The first one acts as a ground where the negative polarity goes, and the other one is where the positive polarity is placed, which we will use to power the board with batteries.

## 6 ANDROID APP

Generally, smart insole devices, usually send the information via wireless. This communication can be carried out in two ways: Wi-Fi or Bluetooth. The information is sent to an application designed by the company, either Android or iOS.

Normally there is a tendency in programming the application more in Android, or for both languages, rather than iOS. This is because more users using Android than iOS. To be able to program in Android, an IDE is needed that allows such programming. Generally, Android Studio is used, which is the official IDE developed by Google.

Using Android Studio, native programming is used, but this is not the only option to be able to program an application on Android. There is hybrid programming that allows the development of a hybrid app on Android. (Griffith 2019a.)

### 6.1 Hybrid app

When talking about a hybrid application, it is also necessary to talk about what a native application is. A hybrid application is a native application.

A native application is one that is programmed from its native code, that is, using the native development language and tools that are specific to a selected platform. An example of that is when using Java compiled by Android Studio to program an Android application. (Griffith 2019a.)

Likewise, a hybrid application is developed in the same way as a web, using the same technologies that are known for it, such as HTML, CSS, and JavaScript. Later the webapp programmed with the mentioned technologies is encapsulated with a native application. This encapsulation gives the hybrid application to be capable of being executed by mobile devices. To carry out the encapsulation, the Apache Cordova is used, and its concept and use are explained later. (Ziglar 2014.)

A hybrid application is a mix of a native application and a webapp. Using a framework like Apache Cordova, the webapp is encapsulated in a native application resulting in a hybrid application, that can be run in mobile devices.

When comparing both types of applications, considering the work involved in programming an app, the hybrid offers faster, easier, and cheaper development compared to the native one. However, a hybrid app is not as fast as the native one. (Griffith, 2019a.)



## 6.2 Angular.io

Angular is one of the most famous and widely used frameworks by all programmers. Is an open-source development platform and framework for creating good quality single-page apps based in Typescript, created by Google. (Google 2010.)

Angular is a framework that implements the MVC pattern (Model, View, Controller). It facilitates the creation and programming of single-page web applications (SPA websites).

## 6.3 Ionic Framework

Ionic is an open source, SDK, focused on the frontend user experience, for building high-quality mobile and desktop apps. It uses web technologies as CSS, HTML and JavaScript. (Ionic 2020a.)

Ionic framework uses Angular as one of the frameworks to be compatible with. This compatibility makes Ionic to take advantage of all the functionalities that Angular has.

To use it, the following command needs to be run (Figure 17):

```
> npm i @ionic/angular
```

Figure 17. Command to install the Ionic Framework based in Angular (Ionic 2020b)

This command will combine the experience core from Ionic with the tooling that is related to Angular. (Ionic 2020b).

The use of a new framework as Ionic is not a difficult or higher-level challenge since you already know how to program in Angular.

One of the advantages to mention of this framework is the cross-platform capability it has. With the same code, you can make different deploys to different devices and / or platforms such as iOS, Android or Windows, without changing or editing the code.

When designing a website, not everything is the functionality. It is necessary to create or program a creative, interactive, and intuitive design that facilitates the use of this app to the client. The design helps the user to not having to try to decipher anything or to waste time looking for some functionality of the app. For this, Ionic provides us with a series of materials or libraries or UI components, for the easy and fast development.

Also, Ionic has its own command line interface to use all the commands that integrates the framework. This is called Ionic CLI (Figure 18), and to install it is necessary to use this command:

```
> npm i @ionic/cli
```

Figure 18. Command to install the Ionic command line interface (Ionic 2020c)

If the command line interface is not installed, it is not possible to execute commands such as the deploy of the programmed code to Windows, iOS, or Android.

To create the project, the following things must be done:

First, the command (Figure 19) is required to install Apache Cordova:

```
> npm i -g native-run cordova-res
```

Figure 19. Command to install Apache Cordova framework (Ionic 2020c)

This command is used to be able to run native apps on different devices and with the Cordova to generate native apps and splash screens. After that, the project needs to be generated (Figure 20) with the following command:

```
> ionic start myApp tabs
```

Figure 20. Command to start a new Ionic project (Ionic 2020c)

The command above, has several parameters, one of them is the name it will receive in the app, in this case "myApp" has been set as the name of the app. Also, "tabs" is another input of the command as it refers to the starter template. Ionic offers three possibilities as starter template that are:

- tabs
- sidemenu
- blank

The next command is used to deploy the webapp into a localhost server, hosted by the computer itself (Figure 21).

```
> ionic serve
```

Figure 21. Command to run with the Ionic CLI our Ionic project (Ionic 2020c)

The good thing about Ionic and its deployment is, that is all dynamic, meaning that whenever a change is made to the code, it upgrades automatically.

## 6.4 Apache Cordova

Apache Cordova is an open source framework focused on mobile application development. It transforms all HTML, CSS, and JavaScript into full-fledged native apps, that can be run in Android or iOS. (Griffith 2019b.)

It contains the necessary tools to transform the code into iOS, Android, and Windows Phone.

To be able to run the application on a different platform other than Windows (Figure 22), it is necessary to use the next command:

```
> ionic cordova run android
```

Figure 22. Command to run our hybrid app, in an android device with Apache Cordova (Ionic 2019)

Where android is the parameter which the app is going to be run. If you want to execute it in iOS, changing the parameter from Android to iOS will work.

## 6.5 Bluetooth Low Energy

Before talking about low energy bluetooth, a talk about what bluetooth is, how it works, and what it offers is needed.

Bluetooth is a protocol for wireless data and voice communication between short range devices. It is integrated into mobile technology devices, computers, and even medical devices. It is because of this reliable communication that the use of Bluetooth has extended and consolidated over the years in everyday tasks. (Bluetooth 2020a.)

Bluetooth technology wirelessly transmits data and voice via radio waves operating in the 2.4 GHz ISM band. It does this by using Wireless Personal Area Networks (WPAN). There is no need of the devices to be aligned when the radio frequency transfer takes place. (Bluetooth 2020a.)

Bluetooth is a standard network that works on two levels:

- On a physical level, as it is a form of radio frequency. It serves to communicate wirelessly with some device.

- At the protocol level, where both devices that want to connect must agree. While one of them sends the information, the other receives it, or the amount of information (bits) that is sent in each wireless transmission via Bluetooth.

In order to make a wireless connection via Bluetooth, both devices (the sender and receiver) must be within radio range.

There are different classes for the different devices that support Bluetooth. These classes vary according to the maximum power allowed and their range in meters from the device.

There are three different class devices:

- Class 1 devices. They have a maximum range of 100 meters and therefore a allowed power of 100 mW.
- Class 2 devices. A range of between 5 and 10 metres, given that their maximum permitted power is 2.5 mW.
- Class 3 devices. They have a maximum power of 1 mW and a range of only one meter.

(Bluetooth 2020b.)

Different standards have emerged during the last years since its creation from 1994.

Starting with the Bluetooth 1.0 / 1.0b standard, followed by the Bluetooth 1.1, Bluetooth 1.2, Bluetooth 2.0, Bluetooth 2.1, Bluetooth 3.0, Bluetooth 4.0, Bluetooth 5.0, and the last one announced a few months ago, which is the Bluetooth 5.1. (Bluetooth 2020a.)

Bluetooth technology supports two radio versions, these two versions are: Bluetooth Low Energy and Bluetooth Classic, since Bluetooth 4.0, as its mentioned above. (Bluetooth 2020b.)

There are a lot of wireless protocols that facilitate the assembly of an IoT network, but what makes Bluetooth Low Energy so tempting, is that it is the easiest way to implement communication between small devices. Being these devices such as Arduino's with sensors or applications on any current mobile platform. (Macías 2017; Townsend 2014.)

The Bluetooth Low Energy (LE) radio is designed for very low power operation. This radio version also supports multiple network topologies, as point-to-point, broadcast, and mesh networking. (Bluetooth 2020b.)

Controlling the connections and the respective ads in BLE, makes our device visible and public to be able to be connected to another one. To determine how they can interact with each other, a profile called "Generic Access Profile" known as GAP is generated. GAP profile objective is to control the connections and the ads. (Townsend 2014.)

This GAP profile allows us to identify and define the different types of roles between the devices. There are two different types of roles: central and peripheral. One of the most important things, is to define who is the device in the central mode, and who is the device in peripheral mode. (Townsend 2014.)

Generally, it is established in peripheral mode, to the small devices, being these of low power. Small devices as they come to be Arduinos boards and sensors that can be connected to more powerful central devices. Regarding the central mode, this is usually applied to devices with more power consumption, being these mobile devices.

There are two ways to transmit information through GAP: The Advertising Data payload and the Scan Response payload. Both payloads are identical, but only the advertising data payload is mandatory. An example of the flow advertising interval can be seen in Figure 25. (Townsend 2014.)

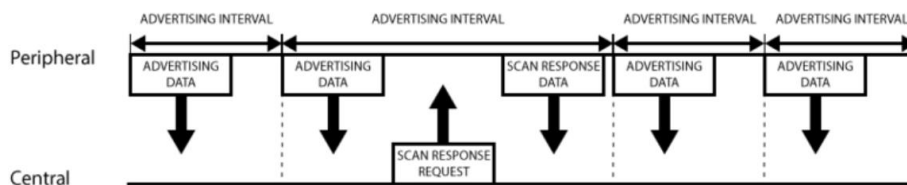


Figure 23. GAP payload example (Macías 2017; Townsend 2014)

High advertising intervals allow you to save battery power, while the short ones allow you to be more reactive. (Townsend 2014.)

When the connection between a peripheral and a central device is established, the advertising process stops. Then the GATT services and characteristics will be used to communicate both devices between them.

The definition of how two BLE devices can communicate using the Services and Characteristics is given by the Generic Attribute Profile under the acronym GATT. GATT comes into play once a connection has been established between two devices, having previously gone through the GAP. (Macías 2017; Townsend 2014.)

It should be noted that that a BLE peripheral can only be connected to one central device (a mobile phone, etc.) at a time. The peripheral device will stop advertising and other devices will no longer be able to view or connect to it, until the existing connection is terminated. Nevertheless, the central device can be connected to several peripherals. In enabling two-way communication and data exchange, it is necessary for the central device to be connected to the peripheral device. (Townsend 2014.)

When it comes to seeing how both the peripheral and the central device exchange information, it can be summarized, in a simple way, as a client-server relationship. Where the peripheral obtains the role of server, being the one that is waiting to receive something in order to work and send an answer, and being the central device, the client that sends those requests to the server. All transactions are initiated by the master device, the GATT Client (central), which receives the response from the GATT Server (peripheral). An example of how the transactions are done, and how the client (central) sends and wait for a response form the server (peripheral), can be seen in Figure 26. (Townsend 2014.)

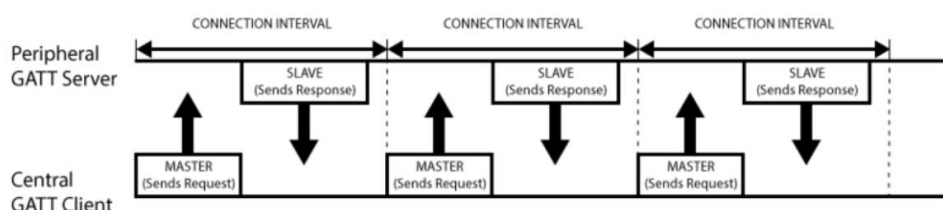


Figure 24. Central-Peripheral GATT communication (Macías 2017; Townsend 2014)

These transactions contain information in every request and response as is logical. Such information is based on high level nested objects called Profiles, Services and Characteristics. (Townsend 2014.)

Services are used to divide data into logical entities and contain specific pieces of data called characteristics. The lowest level concept in GATT transactions is the characteristics, which encapsulate a single type of data. Both are distinguished by a UUID. (Townsend 2014.)

The characteristics are the main element that is used to interact with our BLE peripheral. They can be read-only or write-only and depending of the device, also can notify and write without response of the peripheral. In this way, two-way communications are carried out in a very simple way.

### 6.5.1 Ionic BLE Library

Ionic BLE is the library offered by Ionic to configure the BLE in an Android application, allowing to connect to an Arduino board to receive and send information wirelessly. The library created is also for the Cordova framework as it encapsulates all the web technologies in a hybrid app. (Ionic 2020.)

This library allows to connect to a device, such as sending and receiving information to that connected device, as well as being notified in a specific characteristic (Ionic 2020).

## 7 CASE: SMART INSOLE APP

### 7.1 Smart Insole App

The purpose is to build a real-time system using Arduino. The system will interact with the physical environment. The objective is to build a smart insole device that is capable of giving information about the user's movement, as the user steps.

The system will give extra information of a person's step style, whenever a pressure sensor in the insole detects any type of pressure and the accelerometer identifies any movement between the x-y-z axis.

There are two type of sensors needed, a force pressure sensor and an accelerometer. Both types will be connected to the board of Arduino and will be configured with the Arduino IDE using the C/C++ programming language. (Arduino 2005b.)

Arduino IDE is the software that allows to code the program and upload it to the Arduino board (Arduino 2005b).

#### 7.1.1 Arduino hardware

The first thing was to choose the Arduino board that would be best for the development of the process. Three different models were considered: Arduino Lilypad, Arduino MKR Wi-Fi 1010 and Bluno Beetle BLE.

An analysis was made to see what the final option is. Each board has its advantages and disadvantages.

The first thing was to start with an analysis and decide what would be the minimum requirements that the selected product should meet. The main thing to consider was the size. The board should be embedded in the insole, so it should not be too bulky. The smaller the better, in terms of space and comfort, for this case. Also that size difference should not imply a great loss of functionality in the board, in terms of amount of pins that can be interconnected with the board.

The analysis started with Arduino Lilypad. Because of its size, and because it has a considerable number of pins to make the relevant connections between the sensors and the board, is a good option to consider. Apart from this, a wireless module is needed in order to program a wireless communication between the Arduino and the Android app. The size of the bluetooth module must also be taken into account.

Concerning the Arduino MKR Wi-Fi 1010, it already has incorporated, intrinsically, a wireless connection module. This module supports either Wi-Fi or Bluetooth. With this device, there is no need to buy another component, and this reduces the total size of the product. Its size is slightly larger than Lilypad's, but, on the other hand, a larger size, allows more ports or connection pins on the board.

Analyzing the two previous devices, the ideal device for this case would be a combination of the two, a device of the same or similar size as the Lilypad, but incorporating a wireless communication module. The device found is the Bluno Beetle BLE.

The main aim was looking for a combination of the Arduino Lilypad with the MKR Wi-Fi 1010. In the end, Bluno Beetle BLE appeared as a real option and it is the device chosen for the development of the prototype.

The two main reasons for choosing this device were the following:

- **Size.** Size is a very important factor to consider. As you are developing a device that goes inside a shoe sole, that device should be comfortable to use, and not too bulky and big.
- **Bluetooth.** This device already includes a bluetooth module inside it. This is a significant advantage because it is not necessary to buy a bluetooth module, because it is already embedded. In terms of size is also advantageous, as buying a module it is avoided.

At the same time, the Arduino board is not the only hardware required, there are some sensors that are also necessary, as mentioned in the theoretical part on hardware.

In this case, some force sensitive resistors are needed. For this thesis, the following Force Sensitive Resistor model has been chosen: "FSR07BE - FORCE SENSING RESISTOR 02AH9091" developed by OHMITE. The specifications of the chosen product can be seen in Figure 25, where the last column is for the version FSR07. The size of the sensor FSR07BE, can be seen in Figure 26



CHARACTERISTICS								
Characteristic	Description	FSR01	FSR02	FSR03	FSR04	FSR05	FSR06	FSR07
<b>Actuation force</b>	Force to reach 10M $\Omega$ , Average of 100 samples	< 20g	< 20g	< 10g	< 20g	< 30g	< 15g	< 15g
<b>Force range</b>	linear region of log/log, Higher forces can be achieved with custom sensor and actuation methods	All: Up to 5kg						
<b>Long term drift</b>	1kg for 48hrs, Per log time	< 2%	< 1%	< 1%	< 2%	< 2%	1%	1%
<b>Single part repeatability</b>	100 actuations of 1kg, 1 standard deviation/mean	All: 2%						
<b>Part to part repeatability</b>	100 sensors same batch, 1 standard deviation/mean	All: $\pm 4\%$						
<b>Low temp. storage</b>	-20°C for 250hrs, Avg. change in res. of 5 sensors	8%	7%	7%	8%	8%	7%	7%
<b>High temp. storage</b>	+85°C for 250hrs, Avg. change in res. of 5 sensors	4%	3%	3%	4%	4%	3%	3%
<b>High humidity storage</b>	+85°C/85%RH for 250hrs, Avg. change in res. of 5 sensors	8%	12%	8%	8%	8%	12%	12%
<b>Lifecycle durability</b>	(10M) 1kg force at 3Hz, Avg. change in res. of 4 sensors	17%	12%	3%	7%	7%	3%	3%
<b>Hysteresis</b>	100 actuations of 1kg, Avg. change in res. of 100 samples	All: 5%						
<b>Operational temp. range</b>	100 cycles at 0.5kg	All: -20 to +85°C						

Note: All values typical, and quoted at 10N applied force unless otherwise stated. Force dependant on actuation interface, mechanics, touch location, and measurement electronics.

Figure 25. Ohmite FSR Characterisitcs (Ohmite 2020)

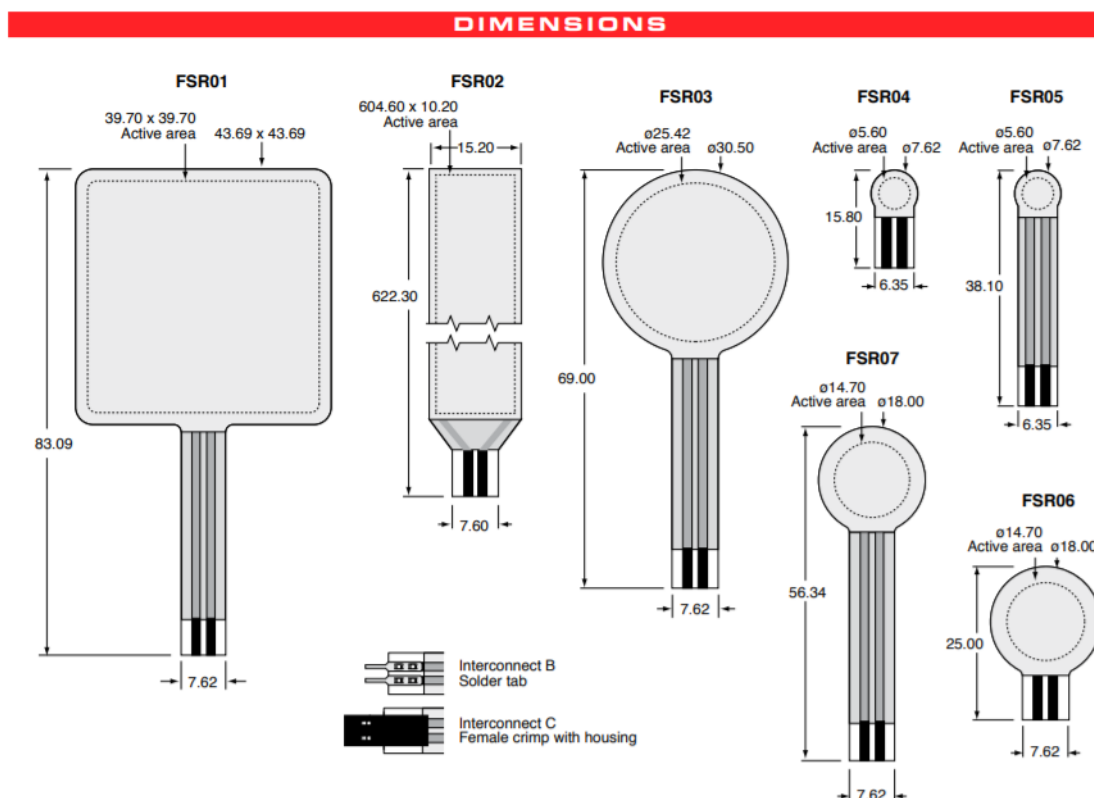


Figure 26. Ohmite FSR Dimesions (Ohmite 2020)

In addition to the force sensitive resistors, an accelerometer for detecting movement is also needed. The digital accelerometer ADXL345 was chosen. It is a small accelerometer that consumes very little, capable of detecting a variation in the three axes (x-,y-,z-). The main reason for choosing this sensor is that it is very standardized and used for mobile applications. It can detect both types of acceleration forces: static and dynamic.

This sensor is compatible with both Arduino and Python by installing the specific libraries needed for its operation (Adafruit 2013).

By using the ADXL345 produced by Adafruit, there will not be any incompatibility problems in the accelerometer as it operates at both 5V and 3.3V. There is no need for a Level Translator Breakout. The accelerometer includes a level shifting and regulation circuitry, allowing to power up the accelerometer with 5V, as the Bluno operates with 5V (Adafruit 2020).

For the prototypes, the use of a LED has been chosen to facilitate the understanding of the logic of the code.

Finally, as far as the hardware is concerned, it is necessary to use resistors to interconnect the sensors with the Bluno Beetle BLE, as well as with the LED.

To calculate the resistances, Ohm's Law was used. With the specifications provided by the LED, a necessary resistance value of  $310\Omega$  was obtained. In the force sensitive resistor case, using a voltage divider circuit, a resistance of  $10K\Omega$  is enough to obtain the data given by the sensors.

### 7.1.2 Arduino software

The prototypes have been made with the Arduino Uno, since the Bluno did not arrive yet. Using Arduino IDE, the board will be programmed. The Arduino IDE version used is 1.6.13. Although the IDE already has a version of 1.8, different problems have arisen with that version. The solution was to use a version earlier than 1.8.

The different libraries used were the following:

- FreeRTOS library
- ADXL345 library
- Adafruit sensor library
- Wire library

Both the libraries of the "ADXL345" and "Adafruit" are used for the accelerometer. To use the accelerometer, it is also necessary to use the "wire" library, which allows us to communicate through the SCL and SDA pins. These pins are directly connected to the accelerometer and Arduino.

A smart insole device is a real-time system. This is because it is waiting for certain events to happen in a certain environment. An event takes place every time the user exerts a pressure on the insole. The system must be able to respond to the event. This response must be correct but must also be delivered before the deadline.

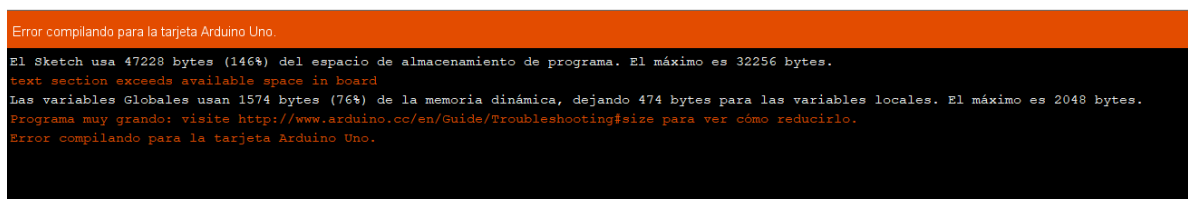
In this case, as it is not a critical system, a soft real-time system is going to be implemented. This type of system allows the response to take a bit longer than the deadline without any problem.

Different tasks need to be handled in the device. The events of pressure on the role and movements must be controlled, and information must be sent wirelessly via Bluetooth between the Bluno and the Android application.

Having more than one task to be running, a task scheduler is needed in order to make context switching. Context switching is needed, as Arduino only supports the execution of one task at a time, due to its architecture. It is necessary to make multitasking with Arduino in order to control the various tasks correctly and efficiently.

To program a real-time system, and at the same time make multitasking programming, the "FreeRTOS" library is used. This library forced using an earlier version, in this case the Arduino IDE 1.6.13 version, than the Arduino IDE 1.8. The reason was that when installing the library, there was a problem. It took up 80% of the sketch, making it impossible to program and declare variables, due to the high memory consumption that the library was using with Arduino.

Later, downgrading the version of the IDE, and reinstalling the library, no longer occupied 80% of the sketch, but 20%. This downgrade allowed the programming in Arduino. The size error can be seen in Figure 27.



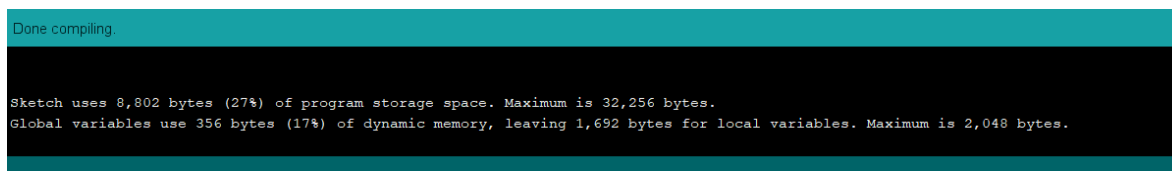
```

Error compilando para la tarjeta Arduino Uno.

El Sketch usa 47228 bytes (146%) del espacio de almacenamiento de programa. El máximo es 32256 bytes.
text section exceeds available space in board
Las variables Globales usan 1574 bytes (76%) de la memoria dinámica, dejando 474 bytes para las variables locales. El máximo es 2048 bytes.
Programa muy grande: visite http://www.arduino.cc/en/Guide/Troubleshooting#size para ver cómo reducirlo.
Error compilando para la tarjeta Arduino Uno.
  
```

Figure 27. Storage size error

With the Arduino IDE 1.6.13 version, that problem is solved. A picture of the output of the sketch can be seen in Figure 28.



```

Done compiling.

Sketch uses 8,802 bytes (27%) of program storage space. Maximum is 32,256 bytes.
Global variables use 356 bytes (17%) of dynamic memory, leaving 1,692 bytes for local variables. Maximum is 2,048 bytes.
  
```

Figure 28. Storage size error solved

After installing the libraries in the arduino IDE, either with the library manager provided by Arduino, or manually including the .zip, it is necessary to include them, in order to use them.

To include the libraries, the "#include" directive is used. This tells the program, in our case the sketch, that some extra code is going to be included from another file. The preprocessor, in this case the Arduino's, reads that line of "#include" and immediately replaces all that command line with the whole library that is included in the sketch. The main function of this directive is to make it easier for programmers to use and handle external libraries.

In order to include the libraries, the following code needs to be in our sketch:

For freeRTOS library, the structure of the "#include" command is presented in Figure 29. In Figure 30 presents the #include command for ADXL345 Library. Finally, in Figures 31 and 32 show the directives for the Adafruit library and Wire library respectively.

```
#include <Arduino_FreeRTOS.h>
```

Figure 29. Include directive for FreeRTOS library

```
#include <Adafruit_ADXL345_U.h>
```

Figure 30. Include directive for ADXL345 library

```
#include <Adafruit_Sensor.h>
```

Figure 31. Include directive for Adafruit library

```
#include <Wire.h>
```

Figure 32. Include directive for Wire library

### 7.1.3 Android software

For the programming of the app, the official code editor of Microsoft has been used, which is Visual Studio Code.

Nevertheless, for the development of the application in Android, Ionic Framework is the framework chosen. The reasons are the following:

- Ionic integrates all the functionalities and API's of the Google framework (Angular), which I had used previously. So, when programming in Angular, you are also using Ionic due to both frameworks sharing the same structure, so the learning of this framework has been simple and relatively fast.

- The simplicity of the realization and exportation of hybrid applications to mobile devices in this case, Android, with the use of Apache Cordova.

Ionic allows by using web technologies already known (HTML, CSS, JavaScript) to program a hybrid application in Android. However, it is also necessary to use the Apache Cordova framework to transform the application into a hybrid app letting any Android mobile device to run it.

The first thought when making the app, was that this app should be simple and intuitive. This is because it only has two characteristics; one is connecting to the board Bluno Beetle BLE via BLE and the other is receiving data transmitted by the Bluno and displaying it.

To make, establish and program the wireless connection between the Bluno Beetle BLE and the Android application, the Ionic BLE library plug-in will be used.

Ionic BLE is a plug-in that enables a communication between a phone and Bluetooth Low Energy (BLE) peripherals. This plugin allows to make a series of functions, which are scanning devices that are compatible and connecting to them. In turn (after connecting), it allows you to receive and send information via wireless communication. The command that needs to be run in the Ionic CLI can be seen in Figure 33.

```
$ cordova plugin add cordova-plugin-ble-central
```

Figure 33. Ionic command to install BLE plugin (Ionic 2020c)

## 7.2 Prototypes

### 7.2.1 Android app prototype v1

The first step of making the prototype was making the Android application since all the necessary physical material for the development had not arrived yet. Before starting, the flow of the application had to be planned.

In order to organize and use the characteristics in a simple and uncomplicated way, a flowchart was made as a guideline for the application. This flowchart of the app can be seen in the Figure 34.

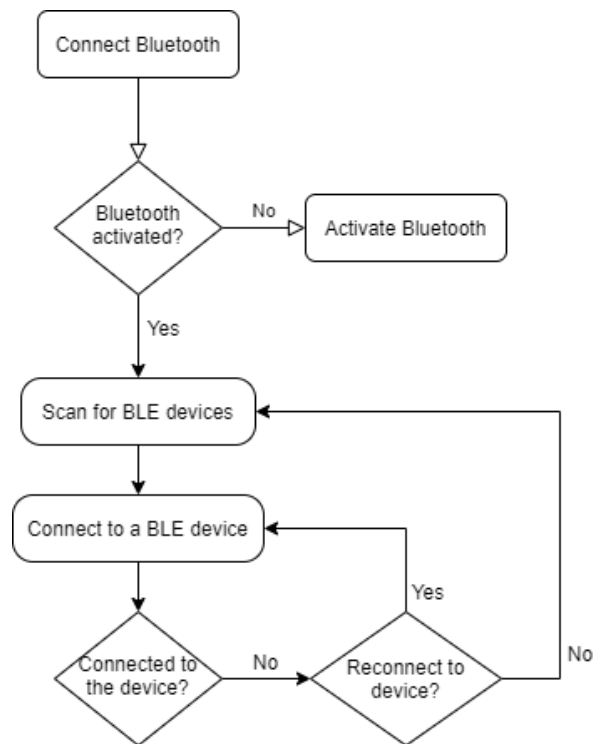


Figure 34. Flowchart diagram of the Smart Insole app

After making the flowchart, it was time to start making the application. The first task was to see how the structure of the code should be, that is, how many views the application should have, and how the navigation should be between those views.

Finally, in this prototype, for all the bluetooth connection, three views were implemented. The views are the home page, the bluetooth page where you scan the compatible devices when connecting, and the view of data.

The different views of the app, which at this point are only functional and have no design in them, can be seen in Figure 35. Looking from left to right, we can find the home menu. This view has a single button to navigate us to the scanning window of bluetooth devices.

The views that has the scan button scans the different compatible devices with which it is possible to make a connection. Then using the connect button takes us to the window of data collection and connection to the selected BLE device.

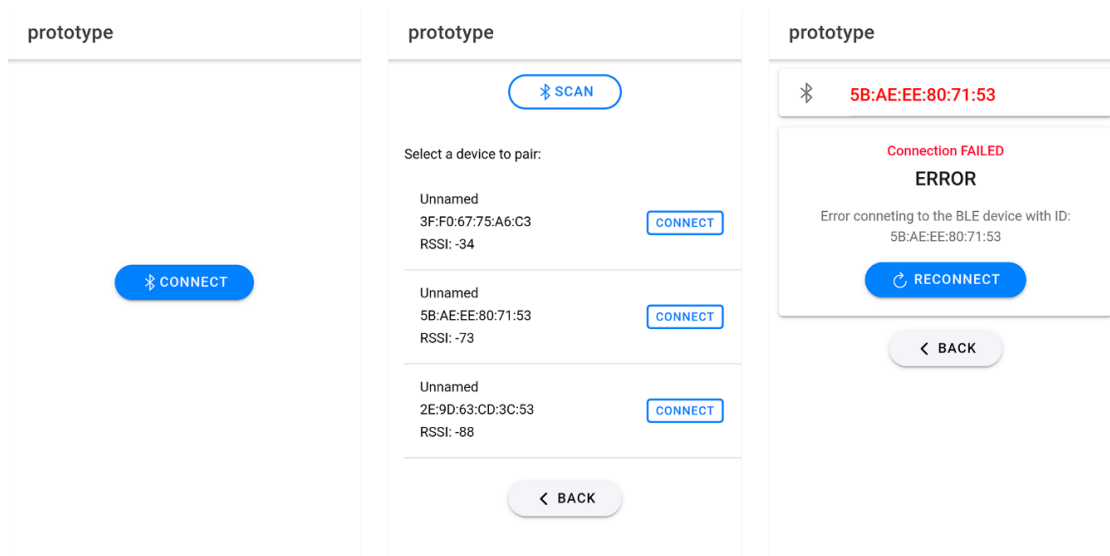


Figure 35. Android prototype app with Bluetooth Low Energy connection and scan done

In this case, there is an error when connecting to the BLE device. This is due to not having all the necessary physical material for the BLE connection. When getting an error, the app itself gives you the option to try to reconnect as it is set in the flowchart.

To scan for compatible devices, the Ionic BLE plugin has been used, as mentioned earlier. It is necessary to initialize it to use it. For the initialization of the plugin, it is necessary to import it and declare it in the constructor of the typescript file. The declaration and initialization of the plugin can be seen in Figure 36.

```
import { BLE } from '@ionic-native/ble/ngx';

constructor(private ble: BLE) { }
```

Figure 36. Import and declaration as well as the initialization of the Ionic BLE plugin

Also, some plugins that make a visual and conceptual improvement to the application were used. In this case, the user will press the scan button. Pressing that button will check if the Bluetooth is enabled or not. If the Bluetooth is not enabled, an informative notification telling to activate the Bluetooth will appear. To do that, the alertController component needs to be imported and initialized (Figure 37) in the same way as with the BLE plugin was. An example of the message can be seen in Figure 38.

```
import { Component } from '@angular/core';
import { AlertController } from '@ionic/angular';
```

Figure 37. Import of the component and AlertController component

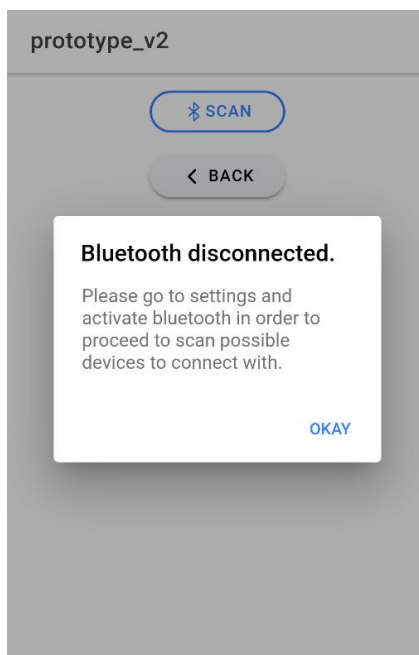


Figure 38. alertController display message due to the Bluetooth being disconnected

Another component has also been used as an indicator that the app is trying to connect to the chosen Bluetooth device. This notification has been done through the component provided by Ionic called loadingController. It is initialized in the same way as the previous component, alertController. An image of the output of the alert can be seen in Figure 39.

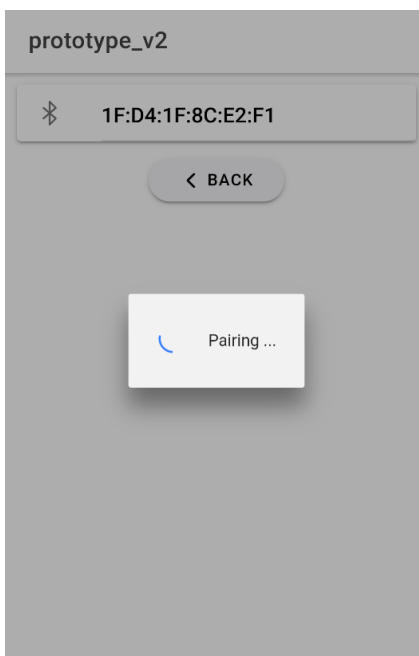


Figure 39. LoadingController output while the app tries to connect with the selected BLE device



The next step will be to check the Bluetooth connection when all the physical material is available. Also, some sketches will be made of different designs of the views to be implemented. These sketches will be made in the free program offered by Adobe, called Adobe XD.

### 7.2.2 Arduino prototype v1

This Arduino part includes the data gathering and processing. The algorithm for detecting a person's steps through the different sensors, is defined in the Arduino part.

The first thing was to know the environment that the real-time system was going to face. The events and how these events could appear are important to program the Arduino respond. The events, in this case, are the pressure on the insole exerted and if there is a movement in the sole.

Analysing the events, is necessary to decide when should the system respond to them. That means if there is a need in implementing a deadline or not. In this case, the Arduino system is chosen to be a soft real-time multitasking system as it faces certain events that are not critical, but an answer is needed.

There was a thinking about how the system should detect what a step is, so that the Arduino reacts to certain events and can give and react to them. Finding the algorithm for detecting a step is one of the key aspects for a proper working of the prototype.

A step is defined as the pressure exerted on all the pressure sensors included and any variation detected in the accelerometer, in the x-y-z axis.

For identifying a step, the Arduino system must be reading data and be attentive to the accelerometer and pressure sensor. These two tasks must be executed at the same time. If pressure is detected, and when checking the variation of the accelerometer it has stopped exerting pressure, there will not be a step. For a step, both movement and pressure exerted must be detected. It is necessary to have both tasks executed at the same time, to be able to identify a step.

Arduino UNO and Bluno Beetle BLE, do not support the execution of more than one task at the same time. Context switching is needed in a multitasking programming due to their system architecture. The architecture only consists of a mono-core CPU allowing a sequential flow.

Doing context switching in a very short time makes the user and programmer feel that both tasks are executed at the same time.



Two tasks are created, one to obtain data (pressure) from the FSR, and another one to obtain data (movement) from the accelerometer.

One of the parameters is to establish a priority of which tasks we want to be executed before the others. In this case, for this Arduino prototype code, it has been decided that both have the same priority. The FreeRTOS task scheduler decides which one goes first.

The first task that is executed. It is within an infinite loop, because this is how a task is defined by the FreeRTOS API. The task will be suspended at each reading that is done from the sensors to do context switching to do the other task. The second task will follow the same structure as the first one. The context switching procedure can be seen in Figure 42.



Figure 42. Output of the Arduino serial monitor, showing one the execution of the different tasks created

As shown in the image above, a multitasking programming is performed. The CPU performs context switching between the ADXL345 task and the FSR task.

At the end, the algorithm to detect the step is as follows. The FSR sensors are being pressed plus the accelerometer detects a variation in the values. In this prototype of Arduino code, a LED turns on when detecting a step. Bluetooth communication is not implemented until Bluno Beetle BLE board is received. (Appendix 1)

### 7.2.3 Testing and conclusions

After making the explained prototypes, the necessary material was obtained to implement the wireless communication.

The first thing that must be said is that the development of the thesis has **not** been possible. This is due to an incompatibility problem between the chosen device, the Bluno Beetle BLE and the library Ionic BLE Library.

The Bluetooth theory and the operation of the Bluno board state that to carry out a Bluetooth communication in a correct way, it is necessary to carry out the following steps:

In the app side, these steps are implemented with Ionic and the Ionic BLE library:

- Connect to the Bluno Beetle BLE.
- After connecting, creating a subscription to the characteristic we are working with, is needed.
- Sending data to the board, which means writing information to it.
- Reading data, due to a notification emerged by the subscription that has been made earlier.

First, in the field of BLE, Bluno offers us two characteristics in which it allows us to send the information, these being: "dfb1" and "dfb2". The first one is used to send the information or to receive from the sensors, while the second one is used only for the BLE configuration.

The "dfb2" characteristic lets the programmer to configure the naming of the device or which role needs to be established for it. The roles can be in central or peripheral mode.

The main problem arose due to the impossibility to subscribe to a certain characteristic. The characteristic offered by Bluno to receive information "dfb1" was the one that cause the problem.

This impossibility arises because each characteristic is defined with a UUID. The characteristic "dfb1" is defined with UUID of 2901. As detailed in the Bluetooth core specification, to subscribe to a characteristic, it is necessary that the descriptor of it is identified by a UUID of 2902. So as "dfb1" has the UUID of 2901, it was not possible to make the subscription (Bluetooth 2020c, 2360).

This impediment in the subscription does not disable the sending of information to the Bluno. It is possible to turn on a LED via Bluetooth by sending a bit. Sending 1 to turn it on and 0 to turn it off. (Appendix 2)

The main impediment that prevents us not being able to subscribe, is when reading the information that the Bluno wants to transmit to us. Subscribing to a characteristic, the app will act every time it detects a change. As it has not been possible to subscribe, the app is not able to detect a change, so it does not know if the Bluno is sending information or not.

If you send information and then the app is waiting for a response to be read, you get as a result what you have previously sent. The app is not able to detect any change because the UUID of the characteristic is not 2902 but 2901.

Making a special approach to the problem arisen, Don Coleman, the creator of the ionic BLE library, suggested a native programming approach as a possible solution for the thesis (Coleman 2020).

The thesis is based in a hybrid programming. If the programming style is changed to native, it allows to program all the connection protocols from scratch, allowing to edit the connection depending on the Bluno BLE settings.

Don Coleman stated that many other people have had the same problem that has arisen. He stated that he cannot change the library because it would affect many other devices causing more problems of incompatibility (Coleman 2020).

Apart from this, it should be noted that the logic of the proposed Arduino code worked correctly, so that it correctly detected a step.

For the Android application itself, scanning for devices and connecting to them worked as expected. It was also possible to send information to the board, as mentioned, but it was not possible to read the characteristic changes made by Bluno itself.

Also, the design of the app was improved. First through the Adobe XD tool, some designs/sketches were made that later were programmed. A readme page is also included, to guide the user in understanding it. All sketches can be seen in Figure 43.

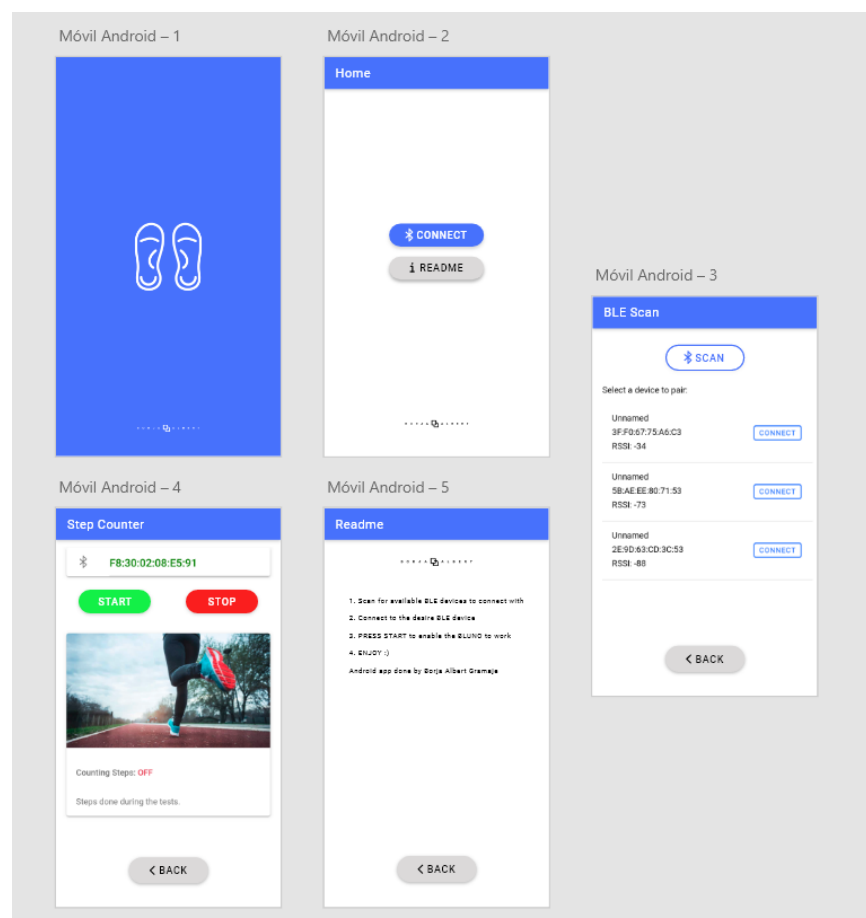


Figure 43. Final sketches for Android app

Even though the Arduino code was not finished, it was thought how it would be the logical code.

As being a real time system (our insole device), it is necessary to know when to give the answer to a certain event in a stipulated programmed time. Considering that the number of steps is not a trivial factor, it was implemented that the Arduino would communicate all the information about the number of steps obtained after the completion of the test. This reasoning was made because the user will not be looking the mobile device all the time as it would not be comfy. For avoiding that, the decision of sending the total amount of the steps in one time was made.

#### 7.2.4 Solutions and future implementations

As Don Coleman pointed out, one way to solve the problem between the Bluno board and the Android app, is to use a native programming. This would have meant a general change in the whole structure of the thesis, as it was focused on hybrid programming. (Coleman 2020)

At the time, it was decided to use a hybrid programming, due to its simplicity and support that both the library and the framework had. However, due to the configuration and attempt of simplify the whole theme of the BLE on the Bluno board, it has been impossible to make the development of the prototype, in the way that was planned.

The focus on native programming, allows a complete configuration of the entire BLE library from the lowest level that there is. This level is the level of code logic. The programmer can configure the code to be compatible with the Bluno board, although it presents the problem of not being able to subscribe to it. Native programming is more flexible because it allows you to program the kind of functionality of the BLE that you want.

Changing the way of programming to the native one, and solving the problem that raised, the next thing would be to implement the BLE connection. Instead of having two tasks as the previous prototype, it would be necessary to implement a third and a fourth one.

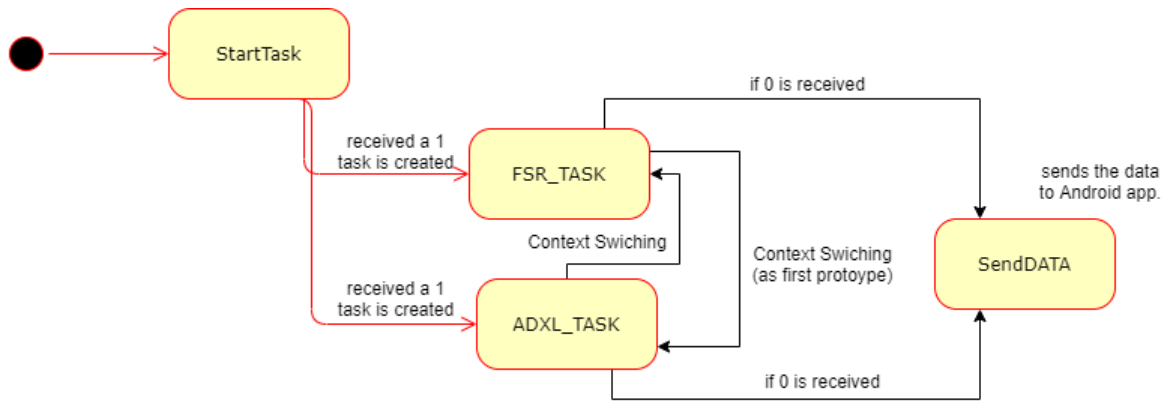


Figure 44. State diagram for BLE and sensors.

The **StartTask** will wait in a loop until it receives from serial a value of one, to indicate that the test is running. When the one is received, **StartTask** creates two more tasks, and it changes its status to blocked. Context switching is made as the first prototype.

As it was decided before, the system would send the information when a zero was received indicating that the tests are done, and the results need to be gathered. So, the deadline period is not as strict as it is in a hard real-time system.

One of the possible implementations that was considered was the development and programming of a heat map. This heat map would show the intensity in which the different pressure sensors were supporting the pressure exerted by the user. This type of information would show the user how he walks.

It was thought to create different work profiles in the app, to obtain data in a precise way, depending on the chosen work profile. Depending on the profile, the person needs to run in different ways. Those profiles could be walking and running.

## CONCLUSION

As far as the technical aspects are concerned, as detailed above, it was not possible to meet the objective of the thesis correctly. This objective was the development of a system for the sole of the shoes, capable of providing information about the steps.

The objective could not be achieved, due to a problem of incompatibilities between the Bluno plate and the BLE library. The library allows the programming of a wireless communication.

A solution was obtained which allowed the development of the objective, but it could not be implemented. It was not realized because the whole implementation should be changed.

Finally, I decided not to implement the solution because the physical products, the last being the Bluno board, took a long time to arrive, leaving me with no time to change the whole programming style.

One of the factors that delayed the arrival of the products was the COVID-19, causing a decrease in reaction time to provide a solution for any problem that could arise.

Also, due to COVID-19, the university was closed, making it impossible to weld the physical components which were required to develop a prototype. All these factors, plus the incompatibility that was found, prevented the development of the objective of the thesis.

On a personal level, in this project, I have learned about all the aspects of IoT, which I had never experienced before. I have found it beneficial and satisfying to be able to investigate and discover new concepts and technologies.

Although it has not been possible to carry out the correct development for different reasons, both for extraordinary reasons such as the COVID-19, and for technological reasons such as incompatibilities between components, it has been beneficial for me to work with this project.



## LIST OF REFERENCES

- Adafruit. 2013a. Overview - ADXL345 Digital Accelerometer [accessed Febraury 25, 2020]. Available at: <https://learn.adafruit.com/adxl345-digital-accelerometer>
- Adafruit. 2012b. Using an FSR Analog Voltage Reading Method [accessed Febraury 3, 2020]. Available at: <https://learn.adafruit.com/force-sensitive-resistor-fsr/using-an-fsr>
- Arduino. 2020a. What is Arduino? [accessed November 12, 2019]. Available at: <https://www.arduino.cc/en/Guide/Introduction>
- Arduino. 2020b. Arduino Software (IDE) [accessed January 9, 2019]. Available at: <https://www.arduino.cc/en/Guide/Environment>
- Arduino. 2020c. ARDUINO UNO REV3 [accessed January 9, 2019]. Available at: <https://store.arduino.cc/arduino-uno-rev3>
- Arduino. 2020d. LilyPad Arduino Main Board [accessed January 10, 2020]. Available at: <https://www.arduino.cc/en/Main/ArduinoBoardLilyPad/>
- Arduino 2020e. ARDUINO MKR WIFI 1010 [accessed January 28, 2020]. Available at: <https://store.arduino.cc/arduino-mkr-wifi-1010>
- Arion. 2017a. About ATO-GEAR [accessed November 17, 2019]. Available at: <https://www.arion.run/about/>
- Arion. 2017b. ARION, Two smart insoles and two footpods for simultaneous measurements [accessed November 17, 2019]. Available at: <https://www.arion.run/product/arion/>
- Barry, R. 2020. Mastering the FreeRTOS™ Real Time Kernel [accessed March 25, 2020]. Available at: [https://www.freertos.org/Documentation/RTOS\\_book.html](https://www.freertos.org/Documentation/RTOS_book.html)
- Bluetooth. 2020a. Understanding Bluetooth Range [accessed April 15, 2020]. Available at: <https://www.bluetooth.com/learn-about-bluetooth/bluetooth-technology/range/>
- Bluetooth. 2020b. Radio Versions [accessed April 15, 2020]. Available at: <https://www.bluetooth.com/learn-about-bluetooth/bluetooth-technology/radio-versions/>
- Bluetooth 2020c. Master Table of Contents & Compliance Requirements [Accessed Arpil 20, 2020]. Available at: <https://www.bluetooth.com/specifications/bluetooth-core-specification/>
- Coleman, D. 2020. Creator of the Ionic/Cordova BLE Library. Interview April 20, 2020.

DFROBOT. 2020. Bluno Beetle - The smallest Arduino Board [accessed February 2, 2020]. Available at: [https://wiki.dfrobot.com/Bluno\\_Beetle\\_SKU\\_DFR0339](https://wiki.dfrobot.com/Bluno_Beetle_SKU_DFR0339)

Digitsole. 2012a. About Digitsole, welcome to the connected footwear world! [accessed November 17, 2019]. Available at: <https://www.digitsole.com/about-us/>

Digitsole. 2012b. Sport Profiler© Running, connected running insoles [accessed November 17, 2019]. Available at: <https://www.digitsole.com/connected-running-insoles-sport-profiler/>

Forciot. 2015a. Company, FORCIOT® story & founders [accessed November 15, 2019]. Available at: <https://www.forciot.com/company/>

Forciot. 2015b. Products and services, unique printed stretchable electronics [accessed November 15, 2019]. Available at: <https://www.forciot.com/technology-and-services/#integrate>

freeRTOS. 2020. Multitasking, RTOS Fundamentals [accessed February 24, 2020]. Available at: <https://www.freertos.org/implementation/a00004.html>

Geeks for Geeks. 2020. Difference between Multiprogramming, multitasking, multithreading and multiprocessing [accessed March 11, 2020]. Available at: <https://www.geeksforgeeks.org/difference-between-multitasking-multithreading-and-multiprocessing/>

Google. 2010. Introduction to the Angular Docs [accessed April 22, 2020]. Available at: <https://angular.io/docs>

Griffith, C. 2020a. What is Hybrid App Development? [accessed April 4, 2020]. Available at: <https://ionicframework.com/resources/articles/what-is-hybrid-app-development>

Griffith, C. 2020b. What is Apache Cordova? [accessed April 22, 2020]. Available at: <https://ionicframework.com/resources/articles/what-is-apache-cordova>

Guimerans, P. 2020. ¿Qué es un sensor? Tipos y diferencias [accessed February 7, 2020]. Available at: <http://paolaquimerans.com/openeart/2018/05/05/que-son-los-sensores/>

Ionic. 2020a. What is Ionic? [accessed January 21, 2020]. Available at: <https://ionicframework.com/what-is-ionic>

Ionic. 2020b. Ionic Angular Overview [accessed January 24, 2020]. Available at: <https://ionicframework.com/docs/angular/overview>

Ionic. 2020c. Your first Ionic App: Angular [accessed January 19, 2020]. Available at: <https://ionicframework.com/docs/angular/your-first-app>

Macías, J. 2017. Bluetooth BLE: el conocido desconocido [accessed April 21, 2020]. Available at: <https://solidgargroup.com/bluetooth-ble-el-conocido-desconocido/>

Mecatrónica Latam. 2020. Que és un sensor? [accessed January 8, 2020]. Available at: <https://www.mecatronicalatam.com/es/tutoriales/sensores/>

Perabo, C. 2016. How do force sensitive resistor (FSR) sensor work? [accessed February 21, 2020]. Available at: [https://www.capling.com/blog/force-sensitive-resistor-fsr-sensor\\_1638/](https://www.capling.com/blog/force-sensitive-resistor-fsr-sensor_1638/)

ResistorGuide. 2020. What is a resistor? [accessed February 10, 2020]. Available at: <http://www.resistorguide.com/what-is-a-resistor/>

Silberschatz, A., Galvin, P. & Gagne, G. 2008. Operating System Concepts, 8th Edition. Available at: <http://www.uobabylon.edu.iq/>

SparkFun. 2020, SparkFun Level Translator Breakout - PCA9306 [accessed April 13, 2020]. Available at: <https://www.sparkfun.com/products/11955>

Townsend, K. 2014. Introduction to Bluetooth Low Energy [accessed April 21, 2020]. Available at: <https://learn.adafruit.com/introduction-to-bluetooth-low-energy>

Ziflaj, A. 2014. Native vs Hybrid App Development? [accessed April 5, 2020]. Available at: <https://www.sitepoint.com/native-vs-hybrid-app-development/>

## LIST OF FIGURES

Adafruit. 2013a. Overview - ADXL345 Digital Accelerometer [accessed February 25, 2020]. Available at: <https://learn.adafruit.com/adxl345-digital-accelerometer>

Adafruit. 2012b. Using an FSR Analog Voltage Reading Method [accessed February 3, 2020]. Available at: <https://learn.adafruit.com/force-sensitive-resistor-fsr/using-an-fsr>

Arduino. 2020b. Arduino Software (IDE) [accessed January 9, 2019]. Available at: <https://www.arduino.cc/en/Guide/Environment>

Arduino. 2020c. ARDUINO UNO REV3 [accessed January 9, 2019]. Available at: <https://store.arduino.cc/arduino-uno-rev3>

Arduino. 2020d. LilyPad Arduino Main Board [accessed January 10, 2020]. Available at: <https://www.arduino.cc/en/Main/ArduinoBoardLilyPad/>

Arduino 2020e. ARDUINO MKR WIFI 1010 [accessed January 28, 2020]. Available at: <https://store.arduino.cc/arduino-mkr-wifi-1010>

Arion. 2017b. ARION, Two smart insoles and two footpods for simultaneous measurements [accessed November 17, 2019]. Available at: <https://www.arion.run/product/arion/>

DFROBOT. 2020. Bluno Beetle - The smallest Arduino Board [accessed February 2, 2020]. Available at: [https://wiki.dfrobot.com/Bluno\\_Beetle\\_SKU\\_DFR0339](https://wiki.dfrobot.com/Bluno_Beetle_SKU_DFR0339)

Digitsole. 2012b. Sport Profiler© Running, connected running insoles [accessed November 17, 2019]. Available at: <https://www.digitsole.com/connected-running-insoles-sport-profiler/>

freeRTOS. 2020. Multitasking, RTOS Fundamentals [accessed February 24, 2020]. Available at: <https://www.freertos.org/implementation/a00004.html>

Ionic. 2020c. Your first Ionic App: Angular [accessed January 19, 2020]. Available at: <https://ionicframework.com/docs/angular/your-first-app>

Macías, J. 2017. Bluetooth BLE: el conocido desconocido [accessed April 21, 2020]. Available at: <https://solidgeargroup.com/bluetooth-ble-el-conocido-desconocido/>

Ohmite. 2020. FSR View Datasheet [accessed March 10, 2020]. Available at: <https://www.ohmite.com/catalog/fsr-series/FSR07CE>

Perabo, C. 2016. How do force sensitive resistor (FSR) sensor work? [accessed February 21, 2020]. Available at: [https://www.capling.com/blog/force-sensitive-resistor-fsr-sensor\\_1638/](https://www.capling.com/blog/force-sensitive-resistor-fsr-sensor_1638/)

Quora. 2017. ¿Para qué sirven todos los controles de la cabina de un avión? [accessed January 18, 2020]. Available at: <https://www.lavanguardia.com/ocio/viajes/20170420/421867243157/que-hacen-controles-cabina-avion-quora.html>

SparkFun. 2020, SparkFun Level Translator Breakout - PCA9306 [accessed April 13, 2020]. Available at: <https://www.sparkfun.com/products/11955>

Townsend, K. 2014. Introduction to Bluetooth Low Energy [accessed April 21, 2020]. Available at: <https://learn.adafruit.com/introduction-to-bluetooth-low-energy>

## APPENDIX 1

In this prototype, the one that worked well, context switching between two tasks is carried out, and there is no BLE connection done. When detecting a step, the system would turn on a LED.

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_ADXL345_U.h>
#include <Arduino_FreeRTOS.h>

//initialize the ADXL345
Adafruit_ADXL345_Unified accel = Adafruit_ADXL345_Unified(12345);
// define tasks for FSR_data & ADXL345_data & LED_task
void FSR_data_task( void *pvParameters );
void ADXL345_task( void *pvParameters );
void Start_task( void *pvParameters );

boolean FSR_step = false;
boolean ADXL_step = false;
int counter_steps = 0;

void setup() {
#ifdef ESP8266
    while (!Serial); // for Leonardo/Micro/Zero
#endif
    Serial.begin(115200);
    pinMode(5, OUTPUT);
    //Initialise the sensor
    if (!accel.begin()) {
        Serial.println("Ooops, no ADXL345 detected ... Check your wiring!");
        while (1); }
    accel.setRange(ADXL345_RANGE_16_G);
    //creating tasks
    xTaskCreate(
        FSR_data_task
        , (const portCHAR *)"FSR_data" // A name just for humans
```

```

, 128 // This stack size can be checked & adjusted by reading the Stack Highwater
, NULL
, 1 // Priority, with 3 (configMAX_PRIORITIES - 1) being the highest, and 0 being the lowest.
, NULL );
xTaskCreate(
  ADXL345_task
  , (const portCHAR *)"ADXL345_data" // A name just for humans
  , 128 // This stack size can be checked & adjusted by reading the Stack Highwater
  , NULL
  , 1 // Priority, with 3 (configMAX_PRIORITIES - 1) being the highest, and 0 being the lowest.
  , NULL );
}

void loop() {
  // put your main code here, to run repeatedly:
  ///no need to put code here as we are using freeRTOS library.
}

/*----- Tasks -----*/
void FSR_data_task(void *pvParameters) // This is a task.
{
  (void) pvParameters;
  pinMode(LED_BUILTIN, OUTPUT);
  for (;;)
  {
    // read the input on analog pin 0,1,2,3:
    int FSR_data_A0 = analogRead(A0);
    int FSR_data_A1 = analogRead(A1);
    int FSR_data_A2 = analogRead(A2);
    int FSR_data_A3 = analogRead(A3);
    // print out the value you read:
    //minimun between 900, in order to detect a higher pressure from the user.
    if (FSR_data_A0 > 900 && FSR_data_A1 > 900 && FSR_data_A2 > 900 && FSR_data_A3 >
900 ) {
      counter_steps += 1;
      FSR_step = true;
    } else {
      FSR_step = false;
    }
  }
}

```

```

}
if (FSR_step == true && ADXL_step == true) {
    digitalWrite(LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
} else {
    digitalWrite(LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
}
//delaying task so, the one that is not executed, goes to executed mode.
vTaskDelay(5); // one tick delay (15ms) in between reads for stability
}
}
void ADXL345_task(void *pvParameters) // This is a task.
{
    (void) pvParameters;
    // initialize digital LED_BUILTIN on pin 13 as an output.
    for (;;)
    {
        Serial.println("ADXL345 TASK");
        sensors_event_t event;
        accel.getEvent(&event);

        //down below, the values are just indicative as it has not been possible to build the whole proto-
        type for choosing more accurate values.
        if (event.acceleration.x > 1 && event.acceleration.z > 5 && event.acceleration.y < 2) {
            ADXL_step = true;
        }
        else {
            ADXL_step = false;
        }
        //delaying task so, the one that is not executed, goes to executed mode.
        vTaskDelay(5); // one tick delay (15ms) in between reads for stability
    }
}

```



## APPENDIX 2

Arduino code where the Bluno receives from the app via BLE information being a 1 or 0. Depending of the result obtained, if its 1, the LED turns on, and if it is 0, the LED turns off.

```
int i = 3;

void setup() {
  Serial.begin(115200); //115200 as the BLUNO specification says.
  pinMode(5, OUTPUT);
}

void loop() {
  while (Serial.available() > 0) //waiting to receive information from the serial via BLE
  {
    i = Serial.read() - '0'; //transforming the char received into a number.
    Serial.println(i);
  }

  if (i == 1) {
    digitalWrite(5, HIGH); //led turn on.
  }

  else if (i == 0) {
    digitalWrite(5, LOW); //led turn off.
  }
}
```