# Mobile Health App for Biofeedback Response in Physiotherapy
## *Development and Validation*

Gonçalo Telo[1,3] and Hugo Gamboa[1,2]
[1]*Faculty of Sciences and Technologies, NOVA University of Lisbon, Lisbon, Portugal*
[2]*LIBPhys-UNL, Lisbon, Portugal*
[3]*PLUX - Wireless Biosignals, S.A., Lisbon, Portugal*

Keywords: mHealth, mHealth App, Biofeedback, Mobile, Physiotherapy.

Abstract: This work consists in developing an electromyographic biofeedback system in the form of a user-friendly mobile application which is simple to use, as it was designed to be an auxiliary component in physiotherapy sessions. This was achieved by implementing a framework that allows the integration of multi-platform plugins, as well as a web view based user interface, which assures the best of the designs allied to the specifications of the native APIs. The communication between the native and the JavaScript methods was tested, as the validation of the application was made internally.

# 1 INTRODUCTION

Bioelectricity is the branch of Bioelectromagnetism that studies electrical phenomena in biological tissues, including areas like Electromyography (EMG) that allow the study of the functioning of muscles and motorneurons(Malmivuo and Plonsey, 1995).

Biofeedback is the information transfer that characterizes a certain state of a biological process, therefore allowing the training of several physiological activities. For this purpose, the level of activity of the organ in study is provided by an electronic instrument, that acquires a signal that is presented to the user in the form of some kind of feedback (visual or sound). This technique can, for instance, be applied to physiotherapy, being able to provide a number of several exercises and a better evaluation of the progression and training of a patient (Bray, 1998).

Surface Electromyography (sEMG) is a non-invasive technique that measures the electric signals from the muscle contraction on the skin, in a phenomenon that is controlled by the nervous system. This signal is composed of the contribution of each motor unit (MU) that activates in order to apply a force and therefore the amplitude of the signal is proportional to the quantity of MUs that participate in it (Kamen and Gabriel, 2009; Chowdhury et al., 2013).

Electronic Health (eHealth) incorporates all the tools and services that use information and communication technologies (ICTs) with the purpose of assisting the healthcare environment either in diagnosis, treatment or monitoring activities (European Comission, 2012). In ICT, parallel development trends such as wireless technologies and ambient intelligence have become some of the most important factors for the evolution of eHealth (Saranummi, 2011).

In 2012, a World Bank report established that 75% of the world's population has access to a mobile phone (Tomlinson et al., 2013). Mobile health (mHealth) covers medical and public health practice supported by mobile phones, patient monitoring devices, personal digital assistants, and other wireless devices (European Comission, 2014).

The creation of an mHealth app that can provide the best of the latest trends both in the health context and in the ICTs is therefore an important step towards the enhancement of medical devices.

This mobile application was developed in a web view basis, which will conjugate the android native (Java) with the web developing languages (JavaScript, HTML, CSS). That option will allow freedom of creation of the Graphical User Interface (GUI) and make sure that it has all the characteristics that we aim to achieve. This means that this is a freestyle mobile application that allows its user to guide his session of physiotherapy independently, with the purpose of enhancing the app's responsiveness and easy adaptation to the several exercises that can be performed.



Telo, G. and Gamboa, H.
Mobile Health App for Biofeedback Response in Physiotherapy - Development and Validation.
DOI: 10.5220/0005825405020506
In *Proceedings of the 9th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2016) - Volume 5: HEALTHINF*, pages 502-506
ISBN: 978-989-758-170-0

# 2 MOBILE APPLICATION

The developed software (physioplux lite) is an electromyographic (EMG) biofeedback system designed for physiotherapists to use both in monitoring and treatment contexts. By providing real-time muscular feedback, it is intended for analysis of the patient's muscular activity which is monitored by one or two medical devices. The latter correspond to muscleBAN devices, wireless electromyographic apparatus, designed and assembled at PLUX - Wireless Biosignals, S.A.. It is a component of a more complex system, but it is also a medical device by itself.

In order to meet the previous description, the app had to fulfil a number of criteria:

1. Provide real-time biofeedback;
2. Help physiotherapists to define goals;
3. Report and track exercise progress, including the primary objective measurements;

In addition, it was also necessary to establish usability and performance requirements, concerning the smartphone's Hardware and Operative System (OS). The list below provides some examples of the selected characteristics:

1. The physioplux lite should be used in Android OS tablets or smartphones with minimum requirements: OS version above 4.3, 1GB RAM, Dual-core 1GHz processor, and integrated bluetooth with touch screen
2. The physioplux lite should be used in its core functions with or without internet connectivity.

Overall, the application can be divided in three main sections: settings, display and report.

In Settings the user selects the device whose feedback he aims to receive. The mobile device scans the local area to find all available muscleBAN devices, providing the MAC Addresses the user can choose from. Depending on the number of devices found, the app automatically defines the use of one or two devices, which reflects on the display. It is also possible for the user to define the name that is shown in the display identifying each device. This should represent the muscle the device is applied to. All the other acquisition parameters are static, in order to assure the quality of the treatment.

The main page is where the data from the devices is displayed. Here the user must define objectives in two steps: firstly, a percentage value is required. This will be used to calculate a threshold, as a result of the selected percentage of the reference value. Then, the user must define if an objective is achieved either by the measured value being higher or lower than this threshold. By default, the reference value is 1 V, however it is possible to establish this by performing a calibration. This is done by measuring the patient's Maximum Voluntary Contraction (MVC), which allows the fixing of the maximum value for each device at the MVC value. As the objective of each bar is met, a goal is achieved and the timer starts counting. This stops when one of the objectives is no longer met. If the new value registered by the timer is higher than the previous, the max value is updated, otherwise it only counts as a goal achieved. Each session allows the execution of as many exercises as the user wants, and a new exercise begins when the user clicks on the exercise display.

The Report is mostly filled automatically. In the report page the user can only define the user's id and his condition, as well as adding descriptions and additional notes to each exercise. The session's starting time and the elapsed time are both displayed, along with the MVC value of each device. Besides that, all of the exercises performed are registered, just as the corresponding thresholds, the number of goals that were met and the maximum time recorded. When the session is finished the user can export the report, as a PDF file, to his email account or to a cloud.

The mobile application can be described with respect to two different areas: the multi-platform plugin and the application itself. In this first phase of development the platform of choice was the Android OS, because of its major market quote. However the app is ready to receive APIs for other operative systems.

## 2.1 Multi-platform Plugin Design

Apache Cordova is an open source framework with a suite of APIs that allow developing mobile apps using JavaScript with access to native device functions. This tool enables the creation of a web view app that is developed just with HTML, CSS and JavaScript in a user interface (UI) framework environment like Intel®XDK. Despite the web technologies that are used, the app is hosted locally (Apache Cordova, 2015).

This software has the enormous advantage of being consistent across multiple device platforms, which permits the migration to other device platforms without considerable changes. This system is available in operative systems such as Android, iOS, Blackberry or Windows Phone (Apache Cordova, 2015).

Developers can also create their own plugin or access the database and use the third-party plugins that the developers community shares, in his own app (Apache Cordova, 2015).

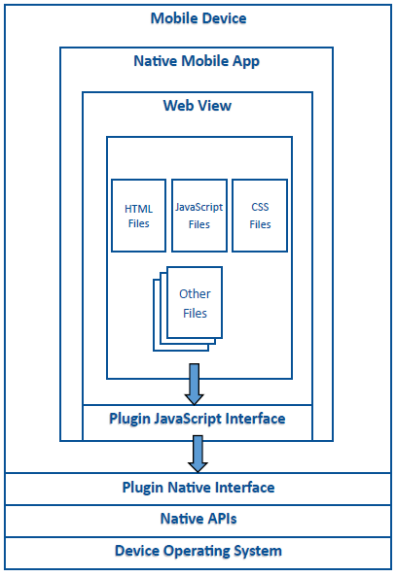


Figure 1: Apache Cordova Native Application Architecture(adapted from (Wargo, 2013)).

#### 2.1.1 Native API

The Native Application Programming Interface (API), is the first level above the OS system layer. This interface enables the communications between our app and the devices.

The communication between the devices and the smartphone/tablet is established using Bluetooth Low Energy. This service results in a lower energy consumption when compared to the classic bluetooth, also guaranteeing a faster form of communication using notifications or indications that fire a callback when the device sends data.

The basic native OS includes methods like connect, disconnect and close. However, in order to communicate with the device we have complementary methods like start, stop and description. Several callbacks are constructed in order to guarantee a good level of communication between the devices, they are represented by data frames, events and command responses.

Data frames are the vehicle used to pass the EMG samples from the device to the mobile application. They consist in a sequence number and an EMG sample value.

Some examples of the events received are: is connected, is disconnected, the battery stage and events concerning the device's correct or incorrect placement. Just as the occurring errors with the command that the API sent to the device represent the responses that can be received.

#### 2.1.2 Plugin Architecture

The plugin architecture is based on the communication between the device's API and the native interface of the Cordova framework, that is called by the JavaScript object that communicates with the top layer of the app.

The native interface is the main core of the plugin. It is where the plugin activity is placed and where the callback to the app interface is set.

There are two main source codes used in the plugin structure. The first is on the JavaScript side and an example of how to call the plugin's native interface is defined as follows:

```
MuscleBan.prototype.connect = function(
                    onSuccess, onError, address)
{
    exec(   onSuccess,
            onError,
            "MuscleBan",
            "connect",
            [address]);
}
```

The next method is what enables the interaction with the device's API by receiving inputs and returning callbacks. The latter reflect either a success or an error case, which are treated by a layer at a higher level.

```
private void connect(JSONArray args,
            CallbackContext callbackCtx) {
    try {
        String macAddress = args.getString(0);

        if(mDeviceService.connect(macAddress)){
            callbackCtx.success();
        } else {
            callbackCtx.error(
               "Error - on try to connect D1");
        }
    } catch (Exception e) {
        Log.e(LOG_TAG, "Error on connect: " +
                                e.getMessage());
    }
}
```

In the matter of data frames or events, the callback channel is open and remains in that state so that the data flow is enhanced.

### 2.2 Application

This web view based application was developed using HTML5, JavaScript and CSS, as well as Apache Cordova plugins like the one described in 2.1. Additionally, we make use of other libraries, such as jQuery mobile, Font-Awesome and jsPDF in order to fulfil the requirements that were planned.

#### 2.2.1 Application User Interface

The UI was designed to provide a user-friendly experience and to be very responsive (Figure 2).

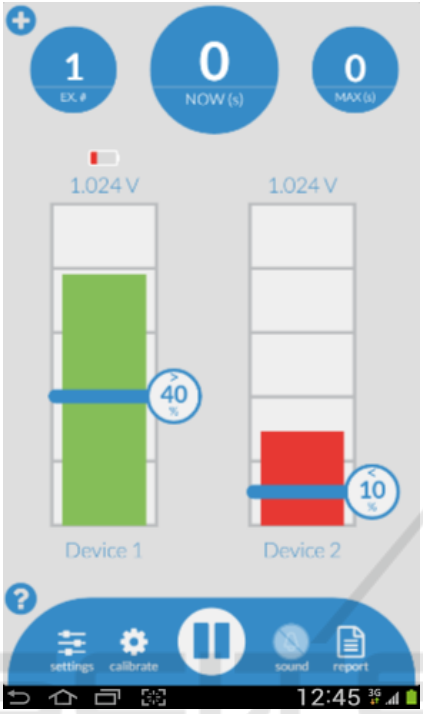


Figure 2: Application's Display Page.

It consists in an HTML5 canvas based interface where all the elements are responsive. The timers (now and max) turn grey on click, with the purpose of not interfering with the session training.

The user can also choose to receive audio notifications in two different modalities: if the objectives are being respected or if the objectives are not being respected.

## 3 SYSTEM EVALUATION

### 3.1 Communication and Visualization Tests

In order to evaluate the readiness of the bluetooth communication with the selected framework, we made a validation test to quantify the delay associated with the biosignal acquisition and its communication through the mobile phone to the user's interface.

For this test we made a simple app, similar to the described in 2, but in this case with two different devices: a virtual one with a square wave behaviour (between zero and max) and a device similar to the ones used to acquire the EMG but with a photodiode. The main goal was for the photodiode to detect the movement of the virtual device bar, counting the time until the response of the stimuli arrived.

The results were very satisfying, with a delay ranged between 100 and 150 ms.

### 3.2 Validation Tests

After the development stage being complete, the app was tested internally by a therapist in a simulated environment. These tests where made using a vertical approach, to assure all the methods and events correct functioning.

Before validating the system in a clinical environment, it was necessary to perform a risk analysis, which is part of the software's technical file. This analysis includes explanations of the several issues that could lead to the misuse of the mobile application and whose severity was classified according to their probability and gravity. This was followed by the mitigation of such problems, as a way of guaranteeing a good quality performance and experience.

The application was also tested by the company's physiotherapist and other engineers, several scenarios and issues where detected and some improvements where made. The most important improvement was the adjustment of the sub-sampling frequency, in order to achieve a good clinical result with a good visual feedback. Other issues concerned the area of the information given to the user, including the critical battery level warning and the misplacement of the device on the skin.

The veracity of the collected data was also taken into consideration during the evaluation, in order to validate the method. This was accomplished by combining the sensibility of the physiotherapist and direct comparisons with other PLUX's products results in the physiotherapy area.

## 4 CONCLUSIONS

The main purpose of this work was to make a robust and steady software, with a user-friendly interface that should allow an easy interaction between the therapist and the patient.

Aiming for this outcome, the whole app was developed from scratch by creating the plugin and the application section, as well as the web view based interface.

The plugin is specific for this app, but it was written so that it can easily be used with other purposes and apps, giving much information to the developer.

The communication between the Java code (native API) and the JavaScript (app code) was found to be responsive and highly reliable.

When the UI design was established it took into account all the good practices, going from the selection of the icons to the colors and the type of events that are displayed.

Many of the decisions that influenced the design of the app, were taken considering the feedback received over time, as a way of guaranteeing the best user experience possible.

As a continuation of this work, we will proceed with the validation tests, concerning the market entry strategy.

The next step to take will be to implement the APIs from other operative systems such as iOS, in order to guarantee the multi-platform compatibility of the app, along with filling the requirements of the two biggest OS in the mobile market.

## REFERENCES

Apache Cordova (2015). Apache Cordova.

Bray, D. (1998). Biofeedback. *Complementary Therapies in Nursing and Midwifery*, 4(1):22 – 24.

Chowdhury, R., Reaz, M., Ali, M., Bakar, A., Chellappan, K., and Chang, T. (2013). Surface Electromyography Signal Processing and Classification Techniques. *Sensors*, 13(9):12431–12466.

European Comission (2012). eHealth Action Plan 2012-2020 - Innovative healthcare for the 21st century.

European Comission (2014). Green Paper on mobile health.

Kamen, G. and Gabriel, D. A. (2009). *Essentials of Electromyography*. Human Kinetics.

Malmivuo, J. and Plonsey, R. (1995). *Bioelectromagnetism: Principles and Applications of Bioelectric and Biomagnetic Fields*. Oxford University Press, New York.

Saranummi, N. (2011). In the Spotlight: Health Information Systems— \vskip-48pt Mainstreaming mHealth\hfill. *IEEE Reviews in Biomedical Engineering*, 4:17–19.

Tomlinson, M., Rotheram-Borus, M. J., Swartz, L., and Tsai, A. C. (2013). Scaling Up mHealth: Where Is the Evidence? *PLoS Medicine*, 10(2):e1001382.

Wargo, J. M. (2013). *Apache Cordova 3 programming*.