Low-power Wireless ECG Monitoring System for Android Devices

Pablo Fernández
Rafael de la Hoz
Miguel Márquez

May 23, 2012

Abstract

Keywords

Acknowledgements

Contents

List of Figures

List of Tables

Chapter 1

Intro

1.1 Project description

As the title suggests, this projects aims for creating a system capable of receiving ECG signals from patients and displaying them so that doctors or specialized staff can analyze them and possibly diagnose heart-related illnesses.

In order to do that, we have been using a system already developed which was responsible of monitoring the patient and sending the resulting data wirelessly as well. These devices were able to send their information through Bluetooth or using IEEE 802.15.4 standard, which is specially relevant for this project, as we will see later on in this document.

Once the results of ECG analysis are emitted, it will be necessary to properly receive and parse them within the displaying device and, finally, show them so they will be human-readable.

This displaying device acts as the system's user interface, and it provides functionality to visualize received data, change visualization parameters, and save and load already-parsed received data also. Thus, considering a monitoring device is sending data, a typical use of the system is: an user executes the developed application on the displaying device, then selects the monitoring device which ECG *he/she wants to analyze and the application starts to show the data as it is receiving it. Meanwhile, that information is being logged to a file at the same time it is being displayed, storing it so as to allow its later, further analysis.

Obviously, regarding the displaying application, there is room to enhance the system's capabilites, as well as to add several useful features. However, such improvements and additions are not considered as they lay out of scope for this project, which focuses in connecting and communicating both devices in addition to translating and displaying the transmitted information.

In short, the main objective for this project, seen as a whole system, consists of visually and wirelessly displaying and managing data obtained from portable, personal ECG-monitoring emitter devices on an Android tablet. In order that the system can operate that way, the following issues would have to be resolved:

1. *Communication with 802.15.4 emitters*

   An initial research was carried out so as to study precedent projects which covered communication using this standard with Android devices, though no results were obtained. Therefore, developing and incorporating this feature to the project would make it pioneer in this field.

   However, this sort of communication is not natively supported by any existing Android device –not even any other widely known portable computing device–, which leads to the following goal for the project to fulfill.

2. *Android accessory development*

   As it was stated before, Android-powered devices are usually equipped with Bluetooth radio modules, yet they lack the capability to communicate with devices which implements other standards. In order to achieve this feature for this sort of device, development of an specifically designed accessory is both a required and mandatory task.

   The aforementioned support Android OS provides for USB device and host modes would allow us to obtain data processed by the accessory through an available interface for almost every Android device. In particular, USB host mode would be required so that the Android device *was able to power the accessory, hence the restriction of using devices running Android version 3.1 or newer.

For this goal an USB-capable board equipped with an MSP430 microcontroller was chosen for acting as the receiver accessory. More specifically, this microcontroller would be running FreeRTOS, which would be accordingly modified for dealing with the USB interface and 802.15.4 communication.

It is also noteworthy that the usage of a prototyping board and a potential miniaturisation of the previously described board were included into the scope of this objective as well. Besides, full description and more details about this development and 802.15.4 communication can be found at chapter 3, Hardware and communications.

3. *Android ECG application*

   Finally, an Android application acts as the system's frontend. Its most relevant requirements were determined by the existing EPFL iOS application, with subtle modifications due to the different platform as well as the inclusion of an extra accessory.

   The data the application displays, in the shape of ECG waves, may be retrieved from a Bluetooth or 802.15.4 streaming node or a local log file. These logs are written by the application itself as it receives an incoming data transmission, so that it can replay them later –original iOS application lacked this feature–.

   Moreover, view controls shall also be offered for the user to modify display density, and move forwards and backwards if a log is being displayed.

   More information about the application, such as requirements and other details, can be found at chapter 2, Software Development.

1.2 Project driver

The main motivation for developing this project was the fact that it meant the gathering of almost every branch of this career. From its very beginning, this work involved both software and hardware development, researching on unknown tools and platforms as well as high and low-level design and programming.

Besides, if successful, it would be likely it could become a real product and be useful both in a professional and particular scope, which added a practical end for the work to be carried out. Something like this could be made thanks to the special features –such as less power consumption and required investment– 802.15.4-compliant technologies provide which wider spread ones lack (Bluetooth, for instance).

In addition, not only developing applications for portable devices but accessories are mainstream nowadays; thus, getting in touch with these activities could provide us with extra experience at leading edge practices, what would broaden our areas of expertise and, consequently, increment our chances to access the labour market.

However, certain areas of the project would mean working on unprecedent techniques –as it will be detailed later on within this section– and dealing with tools which were unknown for us at that moment. Hitches like these could lead to the unfulfillment of the project, yet they could also add extra value to it if they were overcome.

1.3 State of the art

- *EPFL Project*

  As a precedent of the present project, at the École Polytechnique Fédérale de Lausanne (EPFL) a wireless ECG monitoring system was developed, similar to the one has been built up during this project. Just as ours, this system also used *Shimmers as monitoring, transmitter devices; and the obtained data could be displayed in portable computing devices.

  Nonetheless, the list of similarities ends there. The application responsible for rendering the ECG waves was meant to be used over iOS devices, particularly iPhone. This fact led to several additional restrictions, such as mandatory usage of Bluetooth as wireless transmission method as well as the need of “jailbreaking” the device itself. Jailbreak process was needed in order that a explicitly installed Bluetooth stack allowed the device to receive the emitted data properly. Therefore, the employment of Android in our project is partially motivated by this sort of limitations other platforms usually impose.

Our project collaborates with EPFL, from whom we have received feedback as well as hardware and software requirements. In fact, the aforementioned iOS application *settled most of the requirements for the Android one in our project, although there were added some extra ones –such as making logs from received data so they can be read again later–.

- *IEEE 802.15.4*

  This standard describes the physical and Media Access Control (MAC) layers for low-rate wireless personal area networks. It is intended to be implemented into embedded devices, so as to build up short-range networks –10 meters, tipically– with narrow bandwith, up to 250kbps, among other possibilites with lower transfer rates.

  802.15.4 is specially suitable for this kind of project due to its low power consumption. In fact, ZigBee, which uses this standard as its low-level layer, presents a series of advantages over Bluetooth, underlying technology of the previously mentioned EPFL project:

  - **Lower power consumption:** 30mA, ($3\mu A$ idle); while Bluetooth uses 40mA ($200\mu A$ idle).
  - **Bandwidth:** Bluetooth offers much wider bandwidth, up to 3Mbps, meanwhile ZigBee only offers up to 250kbps. This, however, is not a relevant disadvantage because our needs are not so high.
  - **Host number:** ZigBee allows to build networks with up to 65535 hosts, subnetworks with 255 hosts. On the ohter hand, Bluetooth can only support as much as 8 hosts within a network.

  ZigBee, though, is not employed as a whole within this project, but 802.15.4 standard as its basis. Nevertheless, its advantages over Bluetooth remains the same.

- *Android Accessory Development*

  As of May, 2011, there were no easy nor official methods to develop accessories capable of communicating with Android running devices. At that certain time, the release of the Android Open Accessory Development Kit (also known as “ADK”) was announced in San Francisco, within the context of Google I/O, developers conference arranged by Google.

ADK consists of an USB microcontroller board based on Arduino (Arduino Mega2560 to be precise) and a series of software libraries which add specific functionalities and support for other hardware add-ons, tipically known as *shields*, that equip the accessory with sensors or interactive elements which broaden its capabilities. Shields are plugged to the board through its numerous input/output pins, which also allow the connection of personally crafted hardware additions –allowing that way to create tailored behaviours, following the Arduino’s “Do It Yourself” (DIY) spirit.

With the release of that kit, Android project opened itself to the development of all kind of new accessories which would add potential and functionalities it lacked.

As well as this kit, the following release of Android 3.1 API version completed the accessory ecosystem with the inclusion of directly supported host and device USB modes –this support was also backported to Android v2.3.4; only the device mode, though–. By doing so, Google completely cleared the way for the development of Android-compatible accessories, which was previously reduced to the underlying, quite complete but not enough, Linux kernel driver support.

- *ZigBee dongles*

  In spite of the fact that there were available devices which implemented the complete ZigBee stack, they were only designed for being used connected to personal computers, some models with OS restrictions as well.

1.4 Document overview

Over the following pages, this document will present the details about the project it refers to. Beginning with a deep description and analysis of the

Android application's design and implementation, which can be found at the "Software Development" chapter; next, a thorough explanation about the employed hardware and the work it required; and, finally, an exposition of the obtained results, potential expansions and findings.

Chapter 2

Software Development

2.1 Overview

The development of a software application targeted at Android Operating System for mobile devices is the counter-part to the hardware research part of the project. This application was to substitute the already developed one for iOS devices, adding funcionality extracted from feedback obtained from actual medical staff [!]Fran and EPFL[!]. The software must provide functionality to visualize ECG data from Bluetooth or 802.15.4 sources (the latter obtained via [!]our receiver node[!]) in realtime, as well as to save that data into file logs for afterwards reading.

Android as a development platform provides a wide set of high abstraction level tools to emphasize robust and reusable design for low resource based, quick development cycles. Such benefits require the adequation of the software design and architecture to the constrains imposed by the Android development framework.

Given that none of the project team members had received any instruction on this framework, engaging the development of an Android application implied an important risk. Moreover, after the research and training steps concluded, follow up of that risk was not halted, as the quick, robust software development is only assured when building an standard Android application; dynamic, soft real-time functionality implementation is not discouraged, but also not guaranteed to work. Mobile devices development restrictions and common practices were also unkown to the team.

Even when the aforementioned eased development features are applicable, mobile devices are harsh software environments due to, amongst others, memory and battery constrains, where processes have to handle being suspended by an incoming call or similar external events. This factors are specially critical for an application as the one developed in this project, which needs to continually parse and log data.

The application was also intended to act as a quick testing front-end for the prototypes produced by the parallel-conducted hardware research. By providing fully-functional application modules since early stages of development, hardware prototypes could be best-case and worst-case checked by directly connecting them to the Android device for data visualization. Visual verification proved to be a very effective method when working with large quantities of data which were more easily checked against their visual representation than value-by-value reading.

These factors lead to the adoption of an agile software development process focusing on functionality building while prototyping more high risk involving features. To avoid typical drawbacks of such methodologies, great emphasis was put on the application of characteristics found in *Iterative and Incremental processes*, namely, use case driven and risk focused development. That way, project scheduling was done addressing higher risks first while assuring expected functionality to be implemented on time thanks to the use case model.

In the following sections a complete view of the software development project will be presented, beginning with the requirements captured for the project. The use case scenarios identified from those requisites will be detailed next, followed by an explanation of the system architecture ?via 4+1 view model?. Then implementation details will be exposed and the chapter will finish with a short conclusion.

2.2 Requirements

2.2.1 Functional Requirements

- R01 - Receive raw data via Bluetooth
- R02 - Receive raw data via 802.15.4
- R03 - Receive raw data from a log file

- R04 - Parse raw data into processed data
- R05 - Display processed data
- R06 - Log raw data
- R07 - Log processed data
- R08 - Scale View Vertically
- R09 - Scroll View Vertically
- R10 - Scroll View Horizontally

2.2.2 Non-functional Requirements

The following non-functional requirements are identified:

- The application must display ECG data at 30fps.
- The application must run on a Motorla Xoom device.

2.3 Risk Analysis

Being the project mainly a hardware research project, and considering the software development part of it useless without successful results on the hardware part, a detailed process of risk analysis was mandatory to be conducted since the earlier stages of planning and development so as to avoid wasting manpower on futile work.

The risk list at the end of the project is as follows:

- **PR1.** Application funcionality inferior to that featured by existing iOS application
- **HR1.** 802.15.4 receiver device delayed
- **HR2.** 802.15.4 receiver device unfeasible
- **AR1.** Lack of instruction on Android development delays workflow
- **AR2.** Android providing subpar performance when handling required data
- **AR3.** Android rendering capabilities unable to handle required data

- **MR1.** Mobile device unsuitable for target functionality

This risk anaylisis focused on two main risk sources: the parallel-conducted hardware research, and Android as a development platform. Project definition and team related risks were also considered.

The hardware research part of the project delivered the highest probability and impact rated risks. It was so because those risks were external to the software development project scope and thus could not be handled by any of the tools provided by any development methodology. At the same time, should such risks come to be, the impact on the software product would be, in most of cases, as cathastrophic as turning the whole development useless thus causing it's cancellation.

Regarding Android development only a subset of the final set of risks was assessed at first. Every risk in this subset dealt with the team lack of knowledge about the Android platform and was scheduled to be addressed foremost. A last risk was added to this group after the first research on mobile devices limitations regarding potential unfitness of such devices for near real-time display and handling of not-so-small data packages, and that risk handling plan proved to be key to the successful outcome of the project as the remaining subset of Android-related risks were linked to Android applications display performance.

The usual project definition and personal risks such as incorrect deadline scheduling or unability to reach critical milestones on time were pondered, increasing their impact rates as the application would be needed by the hardware device to secure a successful outcome for the project.

A detailed view of each assessed risk is provided next, including risk evolution throughout the project lifetime.

PR1. Application functionality inferior to that featured by existing iOS application
Probability: Moderate
Impact: Very High
Description: Failure to provide an expanded set of features in the Android application when compared with the iOS application renders the software part of the project invalid on its own. It could, then, only be valid as demo software for USB receiver device showcasing. If the device is nor finished,

then the whole software development project will have been futile. The key marker for this risk is unability to generate valid software modules throughout the development that provide required functionality. Failure to reach milestones and use case realizations on time is other important marker. Preventive measures were taken to avoid the occurence of this risk since the beginning of the development by a functionality building focused project scheduling for the first development phases.
This risk was marked as surpassed at the reviewing metting of Iteration 2 as all key functionality had been implemented, as planned.

HR1. 802.15.4 receiver device delayed
Probability: High
Impact: High
Description: Being the production of the 802.15.4 receiver device dependant on the hardware research part of the project a delay on the estimated milestones for that part of the project is likely to occur. Should that happen, hardware research and development will need to be prioritized over this software project. That could lead to big delays in software production. To prevent the rising of further problems derived from those potential delays, the software development process must always work with non-solid, ready-to-change deadlines and milestones. Application functionality is to be ranked in order of importance of implementation to be prepared, in case of an unexpectedly big delay, to leave less important functionality out of the scope of the project. Markers to be followed up are: unsuccessful output from hardware research (a new branch of the potential technologies tree has to be explored), failure to reach hardware development or research milestones and delays in the acquisition of tools or devices needed for the hardware project. Preventive measures considered are: detailed follow up of the hardware research development, reducing the software development team if manpower is needed in the hardware area, and planning asuming delays on component acquisition.
HR1 was monitored throughout the whole software development project, and marked as surpassed at the reviewing meeting of Iteration 5.

HR2. 802.15.4 receiver device unfeasibe
Probability: Medium
Impact: Critical
Description: Until hardware research results are successfully delivered there is no guarantee of the viability of the 802.15.4 receiver device. This software

development project loses most of its value if such device is not developed, as the iOS application already exists. Developing an Android application with an equal feature set is also a valid objective, so this risk does not render the development invalid: the full team will then work on software development, and requirements will be restated to include more final-user oriented functionality and/or features from the *future* set. This risk can be monitored with the following markers: unsuccessful output from hardware research and failure to reach hardware development or research milestones. Being both external to this software project, no preventive measures can be applied apart from scheduling allowing smaller team sizes for the software area.
The probability of the risk was reduced to low after the reviewing meeting for Iteration 3, when the critical hardware research had concluded with positive results. HR2 was marked as surpassed when the production of a device prototype was finished and tested [?TIME?].

AR1. Lack of instruction on Android development delays workflow
Probability: High
Impact: Moderate
Description: None of the team members has received any instruction on Android development and throughout research is not viable because of time restrictions. It is reasonable to foresee potential delays in the development because of the parallel instruction-development flow, as well as the need to rewrite parts of the system rendered obsolete when further knowledge is acquired. Application malfunctioning, unexpected behaviours and low performance are markers to be tracked. As a preventive measure a short instruction time will be scheduled at the beginning of the project, but every team member is responsible to continue his instruction throughout the whole project. Application builds are to be checked for big differences against canon Android applications behaviour.
The risk was marked as surpassed after Iteration 3, as critical functionality had already been implemented and tested, though Android instruction was not halted.

AR2. Android providing subpar performance when handling required data
Probability: Moderate
Impact: High
Description: The benefits of the high-level, single application model provided by Android are such in behalf of the sacrifice of performance. In this

project soft-realtime requirements are present, and the system needs to process around 250 ECG wave samples [quotation needed] (among other data) per second. Android code reutilization and class based programming suggested practices, the absence of an explicit memory management API and the employment of the Garbage Collector only complicate the achievement of such requirements. Special care will need to be put on the development and performance checks are to be conducted regularly on generated builds to ensure the avoidance of this risk. If evidence is found of Android inability to provide the required perfomance (and there is no way of attributting the failure to the team's lack of ability), low-level Android development will be considered. As the probability of this last scenario to occur is quite low, no research will be conducted in low-level Android development until mandatory.
The risk was verified to be occuring during Iteration 2 testing phase. Lack of care on memory management was found to be the problem, and was solved in Iteration 3. The risk was marked as surpassed after the review meeting for Iteration 4.

AR3. Android rendering capabilities unable to handle required data
Probability: Low
Impact: High
Description: The Android Operating System runs on quite a wide range of devices, each with its own technical specifications. Providing the single application development model that Android features requires many software layers, many of them of high abstaction level. The risk exists, thus, that the rendering required by the project couldn't be achieved within the involved time restrictions. The target device for the project is fixed (see Non-functional Requirements Subsection [link!]) as a Motorola Xoom. This device employs a dedicated Tegra2 GPU [quotation needed] which should suffice, so risk probability is chosen as *low*. Performance tests in the display module are to be conducted, though, so as to make sure that a correct usage of the available resources is being done. If low rendering performance is detected, Android native level rendering API, Renderscript, is to be looked into as a remedy, once the code is assured to be optimized.
Risk probability was increased to *Moderate* during Iteration 3 as low performance was detected but was considered not critical enough to apply the Renderscript solution. The risk was marked as surpassed in the reviewing meeting of Iteration 4.

MR1. Mobile device unsuitable for target functionality
Probability: ???
Impact: ???
Description: Risk explanation goes here.
Risk evolution goes here.

draft Thanks to this a schedule was developed that prioritized risk supressing and the decission was taken to plan only the first two of the five intended development iterations, leaving the other three as drafts to allow them to evolve at par with the uncancelled risks.

2.4 Use Cases

2.4.1 UC1. View data from Bluetooth

Description The user wish to receive and visualize data from a Bluetooth ECG node in real time. He will start the communication, visualize real-time received data and finish the connection once done.

This use case captures requisites R01, R04, R05, R06, and R07.

Preconditions The application is in the main menu screen.

Main flow

1. The user indicates his will to start Bluetooth data visualization.
2. The system prompts for the node to connect to.
3. The user specifies the desired node.
4. The system manages connection to the node. If unable to establish the connection, see AF1.
5. The system shows processed data to the user. Received data is also logged.
6. The user can now adjust view parameters (See UC4)
7. The user chooses to finish data visualization.
8. The system closes active connections and stops data visualization.
9. The system returns to the main menu.

Alternative Flow 1 The system cannot establish connection to the Bluetooth node selected by the user.

1. The system notifies the user about the problem.
2. The system returns to the main menu.

2.4.2 UC2. View data from USB Receiver

Description The user wish to receive and visualize data from an 802.15.4 ECG node in real time. He will start the communication, visualize real-time received data and finish the connection once done. The data from the node will be received via the USB 802.15.4 receiver device.

This use case captures requisites R02, R04, R05, R06, and R07.

Preconditions The application is in the main menu screen.

Main flow

1. The user indicates his will to start USB receiver data visualization.
2. The system asks the user to connect the USB receiver.
3. The user connects the USB receiver.
4. The system manages connection to the USB receiver. If unable to establish the connection, see AF1.
5. The system shows processed data to the user. Received data is also logged.
6. The user can now adjust view parameters (See UC4)
7. The user chooses to finish data visualization.
8. The system closes active connections and stops data visualization.
9. The system returns to the main menu.

Alternative Flow 1 The system cannot establish connection to the USB receiver device.

1. The system notifies the user about the problem.
2. The system returns to the main menu.

2.4.3 UC3. View data from log file

Description The user wishes to read a log file created from a real time visualization session. He will specify the log file to load, visualize stored data and finish visualization once done.

This use case captures requisites R03, R04 and R05.

Preconditions The application is in the main menu screen.

Main flow

1. The user indicates his will to start log data visualization.
2. The system prompts for the log file to load.
3. The user specifies the desired file.
4. The system reads the selected log file.
5. The system shows logged data to the user.
6. The user can now adjust view parameters (See UC4)
7. The user chooses to finish data visualization.
8. The system stops data visualization.
9. The system returns to the file selection menu.
10. The user selects to return to main menu. Else follow from step 3.
11. The system returns to the main menu.

2.4.4 UC4. Adjust view parameters

Description When visualizing ECG data the user wishes to adjust view parameters such as plot vertical scale, plot vertical scroll and plot horizontal scroll.

This use case captures requisites R08, R09, R10.

Preconditions The application is displaying ECG data.

Main flow

1. The user indicates his will to change the vertical scale.
2. The system updates plot vertical scale.
3. The user indicates his will to change plot vertical scroll.
4. The system updates plot vertical scroll.
5. If the displayed data is read from a log file, see AF1.

Alternate Flow 1 The user is able to control horizontal scroll parameter.

1. The user indicates his will to change the horizontal scroll.
2. The system updates plot horizontal scroll.

2.5 Design and Architecture

2.6 Implementation Details

First two iterations planned so as to reach iOS software functionality, next iterations only drafted because dependant on hw research.

2.6.1 Iteration 1

The main objectives for this first iteration were

- the instruction of the team on Android development,
- the lay out of an initial version of the application architecture and
- the implementation of the Bluetooth receiver module.

Learning of android, product disposable. Architecture as a prototype to evolve to (or be substituted by) next versions. By the end of the iteration, UC1 realization was complete but not final. (Time)Not yet hardware development =¿ full team at this =¿ smaller time (ObjD) Bluetooth module fully developed except nice user interface and overall user friendlyness (Extra) Log writing v1.

Results and adaptation of pre-planned it2

2.6.2 Iteration 2

The main objectives for the second iteration were

- the development of the USB communication module and
- the implementation of the Log visualization module.

Log writing improved. USB module done as USB device, valid for arduino and first msp tests Log reading implemented, scrolling and such. Tests results indicate low performance, huge memory requirements. With the basic interface, first Prototype with (except usb host == 802.15.4) full functionality implemented. That nice and all. Shown to masters and feedback applied. On time ?

Took a loong break here for hardware development

2.6.3 Iteration 3

This is the first just-drafted iteration. Starting from the fully functional prototype, architectural and performance fixes were mandatory. Also, given the positive state of the hw research scheduling was done for the rest of the iterations.

The main objectives for this third iteration were

- the redesign of the application architecture,
- the achievement of required performance in data management and
- the scheduling of the rest of the project time.

(Scheduling seems strange as an objective)

Architecture redesigned targeting easy adaptation and versatility inside the scope of the application. Redesigned visualization initialization flow. Performance increased significantly and memory usage reduced THROUGHOUTLY in realtime view. Fixes in modules according to new architecture. Talk about scheduling??

2.6.4 Iteration 4

Final implementation iteration. Final performance increasing fixes and user interface implementation, as well as user-friendliness globally increased.

USB Host here!

The main objectives for this third iteration were

- the implementation of USB host communication,
- the achievement of required performance in rendering and,
- the implementation of user-friendly interfaces.

This ends the implementation phase, user-friendlyness could be better but what gives. Performance left 50-50 because it was already ok (30fps not 60fps). One iteration left, devoted to testing.

2.6.5 Iteration 5

Testing and validation with the real thing.

Hey, us of the future, I hope everything's ok up there!

2.7 Closure

Chapter 3

Hardware and communications

3.1 Overview

The hardware in our project cover a main need, an external device to be able to communicate our android device with a *shimmer through 802.15.4. Than device have to be a little and low powered device that can be conected as device through USB in an android device. Little, because a device that disturb a regular work is not valid at all. Low powered, because if the cost of power the divece is higher than use the stack bluetooth we lose an important adventage of use 802.15.4. Able to be coneced throught USB in our android device because this is the only way to interact with android for a external device. And finally able to be connected as device to elimminate the needed of an extra batery that would have incresed the cost and size of our device.

In order to achieve this ambitious goal, we divide this develop in milestones that will help us to focus our works in more concrete tasks and correctly finish the project.

Before of introduce more infromation about our project we have a section of technologies that will be very usefull to understand all this chapter and that will be referenced many times in other sections

3.2 Researched Tecnologies

3.2.1 Arduino

3.2.2 MSP430

3.2.3 802.15.4

3.2.4 FreeRTOS

3.2.5 USB device & USB host

3.3 Description

The hardware is the center section of the research in the whole project, not because there wasn't more research, but because nobody has researched this areas. At the begining we just know what we want to do but we have absolutely nor idea or clues about how we can do a very important number of our *milestones. In any cases we don't even know if our goals could be achieved, in particular a very important one, we need to connect a MSP430 to a Android where Android acts as host, and nobody achieve this before and therefore there are no information about that in forums or TI official support.

The project suppose a chalenge because it involves every hardware level, form the lower levels as the PCB design of a device to higher levels as develop parts of a SO. *Aqui molaria poner algo más que dos tristes lineas.

Scarap zone.
As we explain in the subsection 3.2.5 *Como se ponen enlaces?* the host rol is assumed by who manage the connection, and device rol by the one who just use this connection to send and recive data and recive energy, for us, this minds a simpler programming in our MSP430 device, which code is harder to develop than android code.

The interaction between MSP430 and android was the most dangerous risk of this develop because there aren't information of any kind because it's not researched. USB is quite not as simple as everybody believes, there are many different protocols that works with USB, and each protocol can be implemented in very different ways, this implies a huge investigation process to find if any of them can be used by us. The final way to communicate both devices will be extendedly explained en section 3.4.2.

We decided to use 802.15.4 instead of Bluetooth for many reasons, like lower consums or aviability to create networks to cover big surfaces. But the more important one is that *shimmer have a very small battery that, with bluetooth just lasts 5 or 6 hours, but with 802.15.4 it colud lasts more than a week. Talking about the delineator device that have to be carried by the patient, the difference between charge the device 3 or 4 times every day and charge it less than once every week is more than significant.

Parrafo para introducir los hitos en el que no tengo ni idea que decir, viva!

3.4 Milestones

3.4.1 Arduino for Android USB Device Comunication

3.4.2 MSP430 for Android USB Host Comunication

3.4.3 MSP430 and FreeRTOS

3.4.4 802.15.4 in FreeRTOS

3.4.5 802.15.4 & USB coexistence under FreeRTOS

3.4.6 MSP430 based device design

3.5 Final Product

Chapter 4

Results

4.1 Final state

4.2 Potential additions and expansions

4.3 Findings

Appendices

Appendix A

Utilities and tools

Bibliography