Low-power Wireless ECG Monitoring System for Android Devices

Pablo Fernández
Rafael de la Hoz
Miguel Márquez

June 5, 2012

Todo list

Abstract

Keywords

Acknowledgements

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Project description

This project exposes the research conducted for the development of an USB 802.15.4 receiver device for Android based systems employed in a fully functional electrocardiogram monitoring system encompassed in the field of in-home healthcare, as well as the development process involved in the realization of this whole system.

Electrocardiogram (ECG) monitoring consists on the capture and interpretation of the activity of the heart [cit.needed] during a period of time, and continuous, remote ECG monitoring has become one of the main applications of wireless body sensor networks. Great interest has arisen recently among industrial and academic research parties in production of low-power, ambulatory ECG monitoring systems.

In order to maximize portability, such systems have started to employ the widely available smartphone devices as frontends, reducing the amount of extra devices carried by the user to just the ECG capturing node. In 2011 the École Polytechnique Fédérale de Lausanne presented a real-time personal ECG monitoring system which displayed data in an iPhone via Bluetooth.

Being inspired by the results obtained by the aforementioned project, this certain endeavour seeks taking the inherently good low cost and low power aspects of that to the next level by the employment of a less energy requiring wireless personal area network protocol, namely IEEE 802.15.4, and the more accessible Android based platforms.

The system provides the user with a visual representation of his/her ECG wave highlighting relevant points of this in order to simplify its comprehension. Information about heart-beat-rate is also displayed. All this data is stored in realtime in a transparent to the user manner for later visualization with emphasis put in easy handling of the generated files.

This functionality is provided by the three devices that are part of the system: the ECG delineation node, the USB 802.15.4 receiver and the Android device through the front-end application. The ECG delineation node is connected to the body sensor network and is responsible for capturing and analyzing the electrocardiogram wave, as well as encoding and wirelessly sending the data. The USB 802.15.4 receiver is plugged to the Android system and manages data reception following the aforementioned protocol. Finally the Android based application is the real-time data decoder and acts as the frontend to the user, managing connections and storing and displaying received data.

Development of the Android application, realization of the USB 802.15.4 receiver device, employment of an already-existant ECG delineation node and successful intercommunication between all these elements are the project goals.

As stated before the Android application is the front-end with which the user interacts with the whole system. It is designed following Android common practices as the objective is for it to be of simple use with a smooth, if not nil, learning curve. The decision to develop the application for Android platforms was made according to three main reasons:

First, the fact that Android has been in the top three of the most used mobile operating systems rankings since 2010 [quote!] added to the recent growth of the availability of smartphone devices [quotequote] makes targeting the Android platform a must when the objective is making the system reachable for as most people as possible.

Second, and following the same line of reasoning, the lower cost, in general terms, of Android supporting devices, or at least the wide range of prices of these, specially when compared to other operating systems supporting devices such as Apple's line of iOS devices [average price w/quote here] is to be taken into account.

And finally the comfortable, high-level development environment available for Android application production [quote] provides enough facilities for anyone interested in expanding or contributing code to Android part the project to do so in an easy way. Moreover, this environment is delivered free of charge and is of an open-source nature and multiplatform [quote], whereas iOS platform development kits usage requires at least the ownership of an specific machine and operating system [quote].

Regarding the ECG delineation node, it has been mentioned that the objective of this project is the employment of an already existing one. That is so because both the delineator node and the body sensor network that captures the data are complex systems and the development of them would be the scope of a full project. Specifically the delineator node obtained in the aforementioned EPFL project [quote], with modifications from a collaboration between Complutense University of Madrid (UCM) and the EPFL [quote], was applied.

This node was originally developed as a ultra-lowpower ECG delineator with the ability to wirelessly send data through Bluetooth protocol [quote]. The collaboration between UCM and EPFL, among other things, provided the node with functionality to send data using IEEE 802.15.4, but only at the level required to do some measurement and estimations [quote]. Actual 802.15.4 utilization was yet to be a reality, but it was an excellent starting point towards the achievement of the current project objectives.

The key part of the system and the most important risk involving objective of the project is the 802.15.4 USB receiver device for Android platforms. It is a necessity as generally Android devices have no support for low cost wireless communication protocols such as 802.15.4, while providing other higher-cost protocols such as Bluetooth or WiFi. Development of an Android accesory providing the required functionality is, then, a must, as the most battery and time consuming operation in the delineator node is the utilization of the Bluetooth stack, as previously mentioned researchs [1] and [2] concluded.

Correctly reference papers

Other projects exist with similar objectives in mind, even if they are not targeted at the field of biometrics and personal monitoring. The reason for not following or employing those is twofold: first, at the time of the beginning of the project those initiatives were either unfinished or stalled, furthermore they were isolated, generally single-man projects with no official backend or guarantees of conclusion. Second, following the line of achieving a low cost, low sized device the Android system is to act as the host device in the USB communication, thus removing the size-increasing battery requirements for the receiver as the host role provides the power in such communications.

Write a conclusion here so it is not the saddest piece of text work ever.

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). It is also expected that technologies based on IEEE 802.15.4 specification like ZigBee will increase their importance in the medium term due to their potential applications in domotics (home automation).

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, the École Polytechnique Fédérale de Lausanne (EPFL) – represented by its Embedded Systems Laboratory (ESL)–, along with the Complutense University of Madrid (UCM),

developed a wireless ECG monitoring system [1], similar to the one has been built up during this project.
Just as ours, this system also used Shimmer™as monitoring, transmitter platform; 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. This process allows the user to gain root access to the OS, and it was needed in order that a explicitly installed Bluetooth stack allowed the device to receive the emitted data properly. However, according to the manufacturer [2], jailbreaking the device nulls its warranty. 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, typically– with narrow bandwith, up to 250kbps, among other possibilites with lower transfer rates. It is a relatively new standard since its first version was approved in 2003, although the following revision was employed during this development, which was approved in 2006 [3]
  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:** without going into detail, it can be stated that Bluetooth requires more power than ZigBee does. For example, Bluetooth is constantly emitting information, consequently consuming power, while 802.15.4 allows to enable the radio only when it is needed. These statements are properly justified and explained at subsection 2.3.4, 802.15.4.

- **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, with the additional one that it does not compromise soft real-time developments as the complete stack does.

- *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") [4] was announced in San Francisco, within the context of Google I/O, developers conference arranged by Google [5].
  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, typically 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 at the time of the start of this project, they were only designed for being connected to personal computers, some models with OS restrictions as well. A typical model of this sort is presented in [6]
  Moreover, even if they were to be compatible with our target device, the only available dongles fully implemented the ZigBee stack, which also made them unsuitable for this project due to the relatively high latency that fact imposed –as it can be read at subsection 2.3.4, 802.15.4, soft-realtime needs required the usage of the 802.15.4 MAC layer on its own–.

- *Communication with Android using ZigBee*
  One year ago, around mid-2011, there were very few projects which were working on this sort of communication, between Android and 802.15.4 radio-equipped devices. Concretely, the only project of this kind we were aware of was one from Texas Instruments, which was stated to be pioneer in this field (as it can be seen in [7]). Despite that, several differences stands between this project and ours. For instance:

  - **USB communication:** Texas Instruments developed its own Android driver, namely "virtual COM port", so they could directly connect their ZigBee Network Processor (ZNP) to the Android device through USB. Instead, Android USB host mode is used in this project in order to establish the connection between the receiver and the device. Because of using this method, USB communication between MSP430 microcontroller and the Android device has to be specifically implemented and tuned.
  - **Android platform:** TI's project employed Android 2.2, while we are using version 3.1 in ours, due to the aforementioned need of USB host mode support this API provides. In addition, received data is used by our Android application, while TI employed ZigBee communication for less specific ends, such as controlling other devices like PCs, lamps or obtaining data from certain sensors.
  - **Wireless communication:** in order that the ECG delineation could be done in real-time (namely soft real-time, as it will be explained later), this project does not use the complete ZigBee stack, but its 802.15.4 MAC layer. This restriction requires the

radio –in particular, TI CC2420 radio module– has to be programmed specifically. Meanwhile, Texas Instruments made use of its TI CC2530 system on chip (SoC), which fully implemented ZigBee stack.

– **USB dongle:** as it was mentioned before, TI directly plugged their ZNP to an OMAP4 Blaze thanks to their driver. However, so that the same result could be obtained, we had to connect the radio, CC2420 module, to the MSP430 board –*nombre de la placa–, which was equipped with a USB interface. Furthermore, miniaturisation of this board was designed afterwards to make it an usable dongle.

In other words, Texas Instruments' project was actually able to build up working communications of this kind by the time ours was starting; all the same, special requirements like real time needs entail additional challenges for this project to overcome.

1.4 Document overview

Over the following pages, this document will present the details about the project it refers to. Beginning with a thorough explanation about the employed hardware and the work it required, which can be found at the "Hardware and communications" chapter; next, a deep description and analysis of the Android application's design and implementation; and, finally, an exposition of the obtained results, potential expansions and findings.

Chapter 2

Hardware and communications

2.1 Introduction

The hardware research and development part of the project objective is covering a main need: production of an external device that enables communication between an Android system and a *shimmer through 802.15.4. Such a device should be a portable and low-powered device that can be plugged via the widely used USB On-The-Go (USB OTG) to a host Android system, acting the device as a slave.

It has to be small sized because of the target application environment: a particular who requires constant, in-home, ambulatory monitorization. In that scenario unobtrusive operation is a main need, and usual life style activity modification is to be minimized. And it has to be low-powered, because were the power cost of application higher than that of the Bluetooth technology, a main adventage of 802.15.4 is lost.

The ability to communicate through USB is required because, at the time, its is the most low battery consuming method to interact with Android powered platforms for any external device[quote here wlan and bt costs].

And finally it should be able to act as the slave in the USB connection to avoid the need of an extra power source for the device that would increase the cost and size of the product.

In order to achieve such ambitious goals, and being aware of the substancial ammount of research involved in this part of the project, the decision was made to adopt a milestone driven development which simplified scheduling and helped focusing on specific tasks while maintaining a global view of the evolution and the objectives of the project.

2.2 Overview

Before diving any further into the development a section describing technologies involved in the research process is presented, as such information will be key to the understanding of the rest of the chapter and will be throughoutly referenced during the exposition.

Then a description of the hardware research and development process is given, followed by detailed explanation of each projected milestone, including objectives pursued, lines of research developed, results of each one and conclusions obtained. Estimation based decision making being crucial for the correct outcome of the project, special care will be put to explain the motivations for each decision made. The chapter concludes with an exposition of the results of the research and subsequent development.

2.3 Researched Technologies

This part of the project involves a lot of terms and references a number of technologies and concepts which are important to be acquainted with in order to achieve a full understanding of the current chapter. These explanations won't be presented interlaced with the rest of the sections due to the unmanageable size they would acquire leading to a loss of focus which can only act against full comprehension of the exposed content.

2.3.1 Arduino

2.3.2 MSP430

2.3.3 Shimmer

2.3.4 802.15.4

2.3.5 FreeRTOS

2.3.6 USB device & USB host

2.4 Description

The hardware related investigation is the main section of the research part of the project. Not that there wasn't any research involved in other areas, but some critical elements of this part had never been researched before. At the beginning of the project specific objectives were established and main milestones elected, but absolutely no clue or direction for most of those milestones was available. Moreover, in some cases the feasibility of the proposed goals was unknown. Specifically the achievement of the very important objective of USB communication between an [TI's] MSP430 and an Android powered device assuming the latter the role of master and acting the former as a slave was something not done before and therefore no information was available even in the Texas Instrument support site.

This whole development and research poses some quite interesting challenges as it involves working nearly at every abstraction layer present on device development, ranging from schematic capture and PCB design to operating system related development. The main issues to be resolved are:

- **802.15.4 radio reception:** data packets are emitted from the monitoring Shimmer^{TM}and are to be received and dealt by the accessory. In order to do so, FreeRTOS has to be equipped with a working implementation of the radio stack, which involves implementing part of ZigBee's MAC layer, described by the IEEE 802.15.4 standard, as it is what the emitter nodes use.

- **MSP430 USB handling:**

- **MSP430-Android communication:**

Over the following lines within the next section, the aforementioned challenges are detailed in the context of their own development stages. In addition, objectives and results of each one of these stages are also presented.

2.5 Milestones

This section is presented in chronological order and with milestones view due to its research content, and also because of this was the planification that we follow, as we have no idea of what troubles can we found or how far we can ¿overtake? we have to plan the following milestones and its ¿contents?
We have a serie of tentative milestones pretty clear since the start of the proyect for this reseach part, like connect the device with android via USB or be able to communicate trought 802.15.4 with a shimmer.
There are a project making a port of the necessary part of MAC layer of 802.15.4 developed over FreeRTOS by one UCM informatics faculty teacher,

m

Joaquin Recasencionamos profesores o no?En caso de que si, referencia al paper, this port work in shimmer, our target delineatior device, and MSP430F5438A, the easier develop of the 802.15.4 in this kind of devices make that the MSP430 becomes the choosen develop platform.

p

reguntar a recas que ha hecho exactamente el del FreeRTOS que usamos, y preguntar si podemos mentarle Unfortunately this concrete device, the MSP430F5438A, since now the old MSP430, have not support to USB communication, and there are no way to adapt it. Then after seek in TI, the MSP430 manufacturers, the final prptotyping platform was the MSP430F6638, the highest performance device of MSP430F66xx family wich ones provides USB suport. This decision concreete the first tentative milestone into definitives

2.5.1 Arduino for Android USB Device Comunication

This first milestone's objectives are:

- Acquire a suitable Android device prototyping environment,
- manage correct communication between Android and a prototype device, and
- develop an application emulating desired behaviour.

The process involved in the procurement of each objective is exposed next.

1. Acquire a suitable Android device prototyping environment Initial research on the subject of Android device prototyping

   When we decide to develop a usb device the first step is look for a device tha provide the USB host libraries to connect to android, because the USB host device is which implements most of comunication, thanks to this we can focus our efforts in just try a comunication with android with not so much worries, the device selected to this end was the goole ADK, that is based on arduino(link).ADK is a device developed by google to help android developers to make his first prototypes of USB connected devices. The google ADK is an Arduino with all the facilities that a developer can expect such as libraries, some shields, examples and a good documentation, in our case we are specialy interested in the USB host comunication libraries.

2. Manage correct communication between Android and a prototype device

   When we reciebe our google ADK(link to an image?) we investigate documentacion and we start to develop a small arduino aplication based on the expamples, paraelly we develop the android USB device part(link) that we be needed to test this part of communication. The initial develop, just to make the first test was a little long, because even been the main parts of the implementation os USB comunication made we need to learn how to use it. Our first attempts was no very productives because arduino was a new develop platform but especially because this was our first try with android USB comunication.

3. Develop an application emulating desired behaviour

   Several days later we achieve to Andorid and ADK view each other, but in our definitive test we discover that the communication was not correct because what anyones send was not recibed equally by the other one. But finally we discover what was going wrong and fix it to view how our first milestone has been ahieved.

Once finished we discover the huge importance of plan this first milestone, that will not be used in then final device, but it was very usefull to train the team about USB comunication in android. Conidering than this milestone was not as easy as we expect, if we had start the research in the second milestone which investigation is actualy used in the device, we probably found a lot of troubles that wasn't trully related with the investigation and it would be higly sealed.

2.5.2 MSP430 for Android USB Host Comunication

This milestone's objectives are:

- Supply MSP430 with USB protocol application functionality,
- research Android USB protocol related functionality,
- get Android system to recognize MSP430 as a plugged device, and
- manage communication between Android and MSP430 via USB.

Unlike the arduino's part where we know what we have to use and how we wil use it, now we found that we just have an Texas Instruments(TI) API to comunicate the MSP430 with windows and our goal. Both MSP430 microcontroller and MSP430 board was new devices because all the existing ones have no USB port, the microcontroller is a MSP430 6638 and the board is a TS430PZ100USB(link to the MSP430 section).

This API contains any simple aplications for windows, to comunicate by certain protocols with external devices or enumeration tools; an extense documentation about the API and about the USB characteristics of its devices; and a huge suite of examples that implements a lot of USB protocols to comunicate the MSP430 whith windows such as Communications Device Class (CDC), Personal Healthcare Device Class (PHDC), Human Interface Device (HID) in traditional and datatype implementations, Mass Storage Class (MSC) and combinations of any of them. At this point we have an inmense amount of information and we don't really know what can be usefull for us.

Initialy in order to test the TI API we check some of the multiple examples, in our case was the CDC examples that be usefull to learn about basic concepts of what we try to do. Also we try to read part of the generic(not dependant of the MSP430 device nor USB protocol)documentation that TI provides. And it didn't start too good, the first test with a example provide in the API didn't work in windows, its target SO.
After a good time of investigation we found that it can't initialize a certain clock, this clock don't seems to be in the board and we there are not much references about it. Finally we found a little paragraph in one API document

that metion the chance of a USB needed clock was not included in any kind of boards and a recomendation of how should be this clock, when we buy a clock of this features we can check that this was the problems, the TI example finally works.
All this tetst was carried out in a virtual machine with windows usually running over other windows, but in a casual situation when this virtual machine was running over ubuntu we discover that this examples not only don't work in ubuntu, somthing that we assume, but in addition ubuntu can't even detect our MSP430, this fact worry us because android is an UNIX based SO just like ubuntu.

When we try to connect our MSP430 running one of this examples mentioned before we obvserve that, as we expect, android also can't detect it. Initialy we think our only posible solution to this important trouble was make or found a unix driver, then we investigate that way.

After much searching we see that there are no drivers for unix made by anyone, the closest driver we find was a MAC driver(that we are not sure that it going to work) and assuming that we will need to do a driver based on it, we consult some cualified personal in this area that said us to forget the idea of a generic UNIX driver and focus in an android driver, but also recommend that keep researching other ways to communicate android and MSP in order to avoid the develop a driver for MSP430 wich can be a very difficult work. This advice and the fact that there are no warraties of the develop of a driver was succesfull, we decide to leave the driver idea and follow investigating. That was a very risky decision because we have spended a lot of time in this way and we don't know if we will find other idea to ahieve this hard milestone.

After a few days of unsuccesfull days of investigation, we find in a forum a small mention in a comment about android actually implements HID protocol. This protocol was also suported by the MSP430 API, and although the information was found in a not very condiable place, we find it enought to put the full team to work in this, ones made the android USB host(link) and others find a HID aplication into the API to load into the MSP430. The second objetive was attenpted first, with this we can check that our android device finally detect our MSP430.

That great news helps the android development team, than in this moment becomes the full team, to succesfully imlements the android USB host communication in our aplication in just a weekend. Android USB host comunication was higly dependant of what device was in the other side of the communication, thus everybody was needed in this hard and delicated part of the develop to investigate the high cuantity of manuals contained in the API in orther to find and implement all this particularities. Finally we luckily discover that now, our android and MSP430 can also comunicate each other trought our aplication.

That was wtih no doubts the harder and more dangerous part of the research, there was a lot of chances to do not achieve our goal and the hard work of all the team was essential. This milestone suppose a very important fact not only in hardware part but in all the project because now, we can more accurately schedule all of the project.

2.5.3 USB in FreeRTOS

This milestone's objectives are:

- Validate the use of FreeRTOS in the new tarject MSP430
- Validate USB API utilization viability in conjunction with FreeRTOS in MSP430,
- correctly integrate USB API into FreeRTOS, and
- manage USB data sending in FreeRTOS.

Before to start the real objetive of this milestone we need to adapt the FreeRTOS main functionalities to the new MSP430, that been a new device was not actualy supported by. This mind the creation of a good number of new clases, most of them was excatly equal to their homonimes for the MSP430 5438A but other needs some little modifications.

The next need in the milestone was port as soon as posible the TI USB API to a task-based SO like FreeRTOS whithout taking too much care about its correction. The introduction into the FreeRTOS was prety problematic because the size of just the the API was near to the size of the FreeRTOS. Plus, there are a very important risk, USB uses a important number of resources as pines or clock that can be also used by the FreeRTOS to another task, specially risky was the clock because both need a clock, but the selected board have 2 clock spots, and taking care in the port all this themes could resolved and everything works fine.

Once it's done our preocupation was how to order all this files in a correct way, because our free RTOS was a multiplatform SO that must work in MSP430 5438A, Shimmer, and now, the new MSP430 6638 where take place this develope. In the first port where the multiplatform ability of FreeRTOS was not a problem we lost this funcionality. This separation of tasks help pretty much in focus the first steps in this milestone. This new functionality have to be included just in the supported platform, the MSP430 6638, without affect the other ones as before. Using the preciding generalization needed to cohexists Shimmer and MSP430 5438A as a guide this port was not too traumatic.

2.5.4 802.15.4 in FreeRTOS

This milestone's objectives are:

- Validate current implementation of 802.15.4 in FreeRTOS in testing MSP430,
- manage connection to the CC2420 radio module to target MSP430 device,
- port implementation of 802.15.4 to target MSP430 device, and
- prepare such implementation for actual usage.

At the beginining of this milestone there are a port of the needed part of the MAC layer of the 802.15.4, that have been tested in just certain conditions, like send of medium lenght packets.

As we are not sure of the right working of MAC layer in our system we decide to test it with the old board and microchip that provides serial port output that is extreamly usefull in the debug of a real-time system like this. This result to not works for a certain problems as although it's able to recibe 802.15.4 packets it's not programmed to it, and the max size packets wasn't recived correctly.

Once the right working of the MAC layer is tested, it's time to port it to the new board and microchip, that implies a lot of troubles because the TS430PZ100USB have no conection to a CC2420 radio module. This mean that a full study of the 100 aviable pins to discover which ones are actually unused by both the SO and USB comunication. The radio modlule alse need a particular kind of pins in some cases that there are no very abundant. With this study and the mapping of pins made(mencionamos a carlos?) board and radio module is sent to be weld.

While the board is available, we addressed the programming the pin mapping for de MSP430F6638 into it's class in the FreeRTOS using as base the MSP430F5438A pin mapping class. This is a particulary delicated code and was carefully developed, because if just one thing is not perfect the radio simply didn't work at all and the potencial error will be hard to discover.

With both, board and codding finished the radio was tested and it didn't even trun on. The answer to this trouble was found in a code examples provided by TI for the MSP430F6638, specifically in the *Universal Asyncronous Reciver/Transmiter(USART) initialization code* that reslut to differ slightly of the old MSP430 USART initialization.

Finally some small changes was done in order to adapt it to our project needs. With this a fully funcitonal MAC layer working on the MSP430F6638 was achived and just the potential coexistence with the USB was on the air.

2.5.5 802.15.4 & USB coexistence under FreeRTOS

This milestone's objectives are:

- Assess conflict-free coexistence of current implementation of both USB and 802.15.4 modules in MSP430, and
- manage sending data received from 802.15.4 via USB.

With both USB and 802.15.4 communication working separately we need to test that they can work together. There are 2 main risks; hardware, because any pines used in 802.15.4 can be used also in USB and software, because the time between a radio interruption and the next radio interruption could be too short to send the data trought USB.

The hardware risk was adviced much before the begining of this milestone, and when we made de pin mapping we keep in mind this risk, thanks that, this risk was avoided.

However the software risk was initialy not avoided because the Shimmer send data paquets too fast and MSP430 can't manage this amount of information to send it trought USB and some packets was lost. A little adjusts was necesary, the packets sent was concentred in the start of the available time slots then, we spaced it, sending the same number of packets but with the same time between packet and packet, filling the whole aviable time slots.

Now, our system was finally able to send trought USB the packets recived in radio with no losses. This achievment was very important before develop and test the final aplication with several real-time restrictions.

2.5.6 MSP430 based device design

- Board exhaustive analysis,
- Capture of the schematic, and
- PCB design and route.

2.5.7 Final Validation and Release Candidate Version Development

- Final validation of final aplications with prototyping hardware
- Final validation of final aplications with final hardware

2.6 Final Product

Chapter 3

Software Development

3.1 Introduction

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.

3.2 Overview

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.

3.3 Requirements

The requirement capture process for the project considered three main stake-holders: the project masters (¿Nombres?), the EPFL[*] and the project team members; and was done in two sessions. The first one produced the basic requirement list which described the system and was used to schedule the earlier development iterations. This was so because of the strong time restrictions this software development project had to cope with. When the critical functionality was achieved and the hardware research was in a suitable state, the second requirements capture session was conducted. By then, the +EPFL representative (Fran) [Murcia Hospital representative?] had shown the state of the development to the stakeholder party of him, and collected feedback. Thus, the requirements produced by this second session were of a more user-oriented nature.

The functional and non-functional requirement lists are presented next.

3.3.1 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

3.3.2 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.

3.4 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
- **MR1.** Mobile device unsuitable for target functionality
- **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

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 re-

viewing 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?].

MR1. Mobile device unsuitable for target functionality
Probability: Low
Impact: Critical
Description: Even when mobile devices technical specifications have increased significantly in the previous years, specially regarding CPU power and rendering capabilities, the soft-realtime requirements of the project in terms of data manipulation and visual representation involve a low probability risk of the application being unsuitable for such devices. The risk probability is decreased by the fact that similar featured applications exist both for iOS and Android powered devices. Even so, the 802.15.4 receiver device USB interface doesn't allow for this risk not to be reckoned. Preventive measures considered: quick prototyping of critical functionality to discard unfeasibility, conduction of performance tests, both rendering and data handling related, in the target device and testing of USB-Android communication as soon as possible in the project schedule.
As performance tests were conducted on early builds of the application, the need of further research on this risk arised as the data handling performance in the target device was low, but not as low as the rendering performance. Thus, this risk was unfolded into risks AR2 and AR3, both related to Android performance when handling the aforementioned tasks. This risk follow up was then halted, as it was no longer needed.

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.

Thanks to this risk analysis the project schedule was developed in such a manner that it prioritized risk suppressing and the decision 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. Use case realization order was also selected based on this risk analyisis, ensuring that higher risk involving features were developed (or at least prototyped) as early as possible.

Use cases for the project are presented next, followed by the system architecture description, to allow a finer understanding of the software development project schedule and related decissions, which will be detailed in the next section.

3.5 Use Cases

The project use cases are now presented following the *use case common style* described by Martin Fowler [Fowler, 2004], as the relaxed template allows for quicker document producing than more detailed descriptions such as those presented by Cockburn [Cockburn, 2001].

Actor listing is omitted in all the descriptions, because of the following implicits:

- The main actor for the use case is the user
- The only other actor is the data emitter node, when applicable. It can refer to the Bluetooth emmitter device or the USB 802.15.4 receiver device.

Please note that the 802.15.4 emitter node is not considered an actor as the communication with such device is handled by the USB 802.15.4 receiver, which actually is the considered actor.

3.5.1 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. User indicates his will to start Bluetooth data visualization.
2. The system prompts for the node to connect to.
3. User specifies the desired node.
4. The system manages connection to the node. If unable to establish the connection, see AF1.
5. The Bluetooth node sends data to the system.
6. The system shows processed data to the user. Received data is also logged.
7. The user can now adjust view parameters (See UC4)
8. User chooses to finish data visualization.
9. The system closes active connections and stops data visualization.
10. 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.

3.5.2 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. User indicates his will to start USB receiver data visualization.
2. The system asks the user to connect the USB receiver.
3. User connects the USB receiver.
4. The system manages connection to the USB receiver. If unable to establish the connection, see AF1.
5. The USB receiver device sends data to the system.
6. The system shows processed data to the user. Received data is also logged.
7. User can now adjust view parameters (See UC4)
8. User chooses to finish data visualization.
9. The system closes active connections and stops data visualization.
10. 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.

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

3.5.4 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. User indicates his will to change the vertical scale.
2. The system updates plot vertical scale.
3. 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. User indicates his will to change the horizontal scroll.
2. The system updates plot horizontal scroll.

3.6 Design and Architecture

Design made targeting easy expansion of the code. Core, domain specific components that require realization in order to work. Model involves: a Manager, a DataSource and a DataHolder (as well as a Data-Parser) and, if visualization is required, the ECGView and a ViewThread derivate. The model needs to be realized for it to work, the Activity is just an engine running an instance of the model. "It follows a factory pattern, but the factory is the Activity."

View Package: Realtime rendering required a special, non full common functionality (slide, scroll, ...) providing View derivate: SurfaceView. It provides a bitmap surface where you can draw whatever you want. Derived into AnimationView with employment of AnimationThread for obtaning a good old while(!finished) {update(); render();} loop. That structure derivates into ECGView which is a domain specific specialization of AnimationVeiw and Dynamic- and StaticViewThread. The later two implement behaviour expected of Dynamic and Static data visualization respectively (more on this on the Data Package) Programming style in this area closer to C than to an OOP language in order to maximize performance. (Talk about intolerable initial performance caused by extreme number of new each step and too much relaying on GarbageCollector?) Data Management Package: Data class acting as a centralized data storage. Provides app-level syncronization too. Different models for static and dynamic data management. Dynamic:

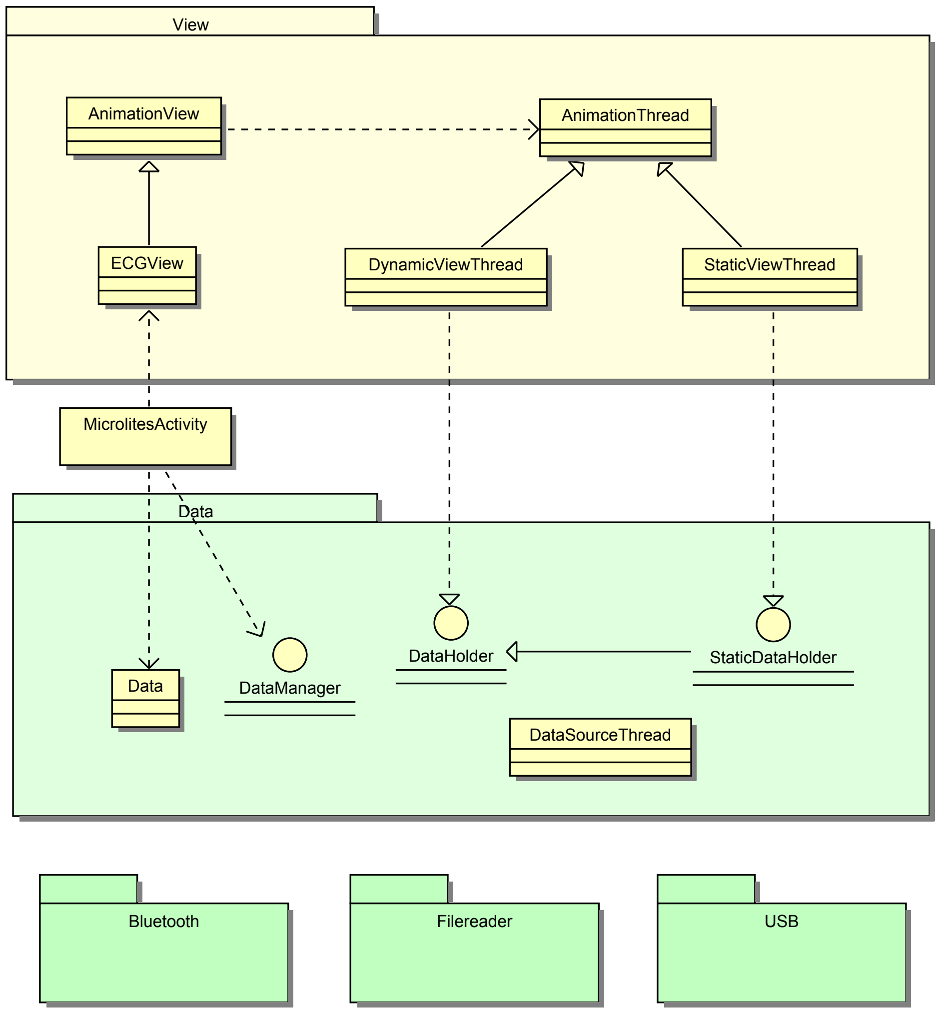

Figure 3.1: Architecture overview

manages its own data and implements replacemente and discard politics, data is provided by single units. This is so because of the way the data is received and then parsed in real-time. Static: obtains a reference to the data location, which is where the actual data manipulation is done. This way huge ammounts of data transfering is avoided (read file can contain lots of megabytes of data which can't be passed at once to the holder). First try was passing the whole array of samples but that was unassumable in real-time.

3.7 Implementation Details

In this section a detailed description of the development is presented. The evolution of the application architecture and functionality; design, architectural and implementation decisions and explanations of the circumstances in which they were made, as well as the development of the project schedule are exposed in chronological order.

The section is arranged following the five iterations of the project, developing each one by presenting the target objectives and expected deadlines for the iteration, detailing application evolution with emphasis on use case realization and risk suppressing achievements and demonstrating the state of the project as assessed on the reviewing meeting conducted at the end of the iteration.

3.7.1 Project Scheduling

Before diving into the description of the development process an overview of the project schedule is mandatory.

As was mentioned in the previous sections, an agile software development methodology has been applied to the project, even if artifacts from more ordered and structured methodologies have been employed to avoid losing focus on the critical aspects of the development.

Software development project being dependant on parallel-conducted hardware research, project assets, both personal and development resources, being shared with the hardware research having the former higher priority, and team's lack of formation regarding Android development are some of the factors which lead to the adoption of such a mixed methodology. As they have been already exposed in previous sections, they won't be detailed here.

In such an scenario, a fully developed, eight month covering schedule was unfeasible, at least with the imposed time restrictions which didn't allow much time to be spent on planning. The decision was then made to plan only the earlier project iterations, assuring critical functionality identified from use case development to be implemented as soon as possible as well as higher threat involving risk suppression to be realized.

Actual software project development begun October 15, 2011, being the final deadline set on June 15, 2012. That deadline could not be pushed any further, so an estimated deadline was set on May 31, 2012, leaving half a month for further work or recovering from delays, and seven and a half months of development time.

Time was needed for the hardware research to start providing results, and no work could be done in the meantime on related application parts. Considering also that even in the most optimistic scenarios no actual work with the 802.15.4 USB receiver couldn't be done until mid February [link to hw planning or something], the schedule had to assume that the first four months of software development would need to cover the implementation of most of the features, leaving the rest of the development time for implementing hardware-related requirements, which couldn't actually be scheduled until hardware research was evaluated.

The adopted schedule dealt with the aforementioned facts by proposing five iterations, from October 15 to May 30, complying with the fifteen days reserved for contingencies. The first three iterations had a duration of two months, the third one a single month, and the remaining one was fifteen days long.

Having the previously developed iOS application as starting point, and being achieving at least the same feature list of that a critical target of the project, the first two iterations were planned so as to realize **UC1** and **UC3** which captured requisites expressing such feature list.

The next iterations were just scrapped as no accurate estimation could be done on the state of the project by those dates. Thus, the third iteration was planned to cover the implementation of the remaining features, namely, communication with the 802.15.4 receiver device, the fourth one reserved for polishing the application and the fifth one left for testing and validation. The fourth iteration was to shrink if objectives were achieved quickly to leave more time for validation.

The scheduled distribution of work for each iteration is as follows:

It	Dates	Activity
1	Oct 15 - Dec 15	Application base and Bluetooth module
2	Dec 15 - Feb 15	Log module and initial USB module
3	Feb 15 - Apr 15	Final USB implementation
4	Apr 15 - May 15	Polishing
5	May 15 - May 30	Validation
-	May 30 - Jun 15	Reserved

Which can be seen as the following expected use case realization dates:

It	Dates	Realized Use Case
1	Oct 15 - Dec 15	UC1, UC4 (first version)
2	Dec 15 - Feb 15	UC3, UC2 (first version)
3	Feb 15 - Apr 15	UC2 (final)
4	Apr 15 - May 15	UC4 (final)
5	May 15 - May 30	Validation
-	May 30 - Jun 15	Reserved

Throughout the development, as expected, this schedule had to be adapted as the hardware research advanced to assess the arise of difficulties and subsequent changes to its planning. Instead of providing a fixed snapshot of the altered schedule at the end of the project, a detailed description of each iteration is presented next, including modifications to the planning and the motivation for making them.

3.7.2 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.

The alloted time for the iteration was of two monts, from October 15 to December 15. During this period the hardware research didn't require much resources [link to hw], so in practical terms the full team was employed in the software project.

In order to minimize risk AR1[quote?link?], the instruction of the team on Android development becomes the key objective for the iteration. Development starts with the implementation of the designed base architecture for the application, so it serves as a powerful learning resource. The notion of that implementation as disposable is not abandoned throughout iteration time and it is always considered as a prototype to evolve or be substituted by a more refined version once team's knowledge of Android increases.

Having the basic architectural framework developed, implementation of the Bluetooth receiver module begins. Special care is put both on design and implementation, as this functionality has to be ready as soon as posible, and little time will be available for reimplementation further in the development.

Testing of both architecture and Bluetooth module is conducted during the whole iteration, specially in the final weeks, so by the end of the iteration Bluetooth module is validated and expected to be stable until the scheduled architectural redesign.

As the Android formation phase takes less time than expected and hardware research doesn't allow advancing into next iteration at the time iteration objectives are achieved, an extra implementation effort is put to begin implementation of UC4. Adjust View Parameters.

In the review meeting for the iteration the following conclusions are obtained:

- Basic architecture implementation is done.
- Android is confirmed to be a very comfortable development environment even if non-standard functionality is not that easily achieved. Instruction is planned to continue.
- Realization of UC1-View data from Bluetooth is finished. Related user interface is not final, but it is functional, and is to be updated on further iterations.
- Realization of UC4-Adjust view parameters is done in relation to Bluetooth visualization. As remaining modules are developed, further work is to be done in this field.

3.7.3 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

3.7.4 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??

3.7.5 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.

3.7.6 Iteration 5

Testing and validation with the real thing.

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

3.8 Closure

Chapter 4

Results

4.1 Final state

4.2 Potential additions and expansions

4.3 Findings

Appendices

Appendix A

Utilities and tools

Bibliography

[1] *Real-Time Wavelet-Based Electrocardiogram Delineation System with Automatic Arrhythmia Detection*
Embedded Systems Laboratory, ESL
École Polytechnique Fédérale de Lausanne, EPFL
`http://esl.epfl.ch/cms/lang/en/pid/46016`

[2] *Unauthorized modification of iOS as a major source of instability, disruption of services, and other issues*
Apple Support
September 27, 2010
`http://support.apple.com/kb/ht3743`

[3] *IEEE Std 802.15.4^{TM}-2006*
(Revision of IEEE Std 802.15.4-2003)
IEEE Computer Society
June 7, 2006

[4] *Android Open Accessory Development Kit*
Android Developers
`http://developer.android.com/guide/topics/usb/adk.html`

[5] *Android Open Accessory API and Development Kit (ADK)*
Mike Lockwood, Erik Gilling & Jeff Brown
May 10, 2011
`http://www.google.com/events/io/2011/sessions/android-open-accessory-api-and-development-kit-adk.html`

[6] *IEEE 802.15.4/ZigBee^{TM} USB Dongle*
Integration, Adaptive Modules
`http://www.adaptivemodules.com/assets/File/integration_802-15-4_usb%20dongle.pdf`

[7] *TI presents World's first platform bringing ZigBee to Android smartphones and tablets.*
Texas Instruments
March 04, 2011
http://www.ti.com/ww/in/news_detail/2011/news_detail_20110304.html