Design of a low-cost wireless NIRS system with embedded Linux and a smartphone interface

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Biomedical Engineering

by

Diogo D.S. Dias
BEng, Fundação Federal do ABC, 2014
Bac., Fundação Federal do ABC, 2012

2015
Wright State University
I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Diogo D.S. Dias ENTITLED Design of a low-cost wireless NIRS system with embedded Linux and a smartphone interface BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Biomedical Engineering.

__________________________________________
Nasser H. Kashou  
Thesis Director

__________________________________________
Jaime E. Ramirez-Vick, Ph.D.,  
Chair, Department of Biomedical, Industrial and Human Factors Engineering

Committee on  
Final Examination

__________________________________________
Jack Jean, Ph.D.

__________________________________________
Ulas Sunar, Ph.D.

__________________________________________
Nasser H. Kashou, Ph.D.

__________________________________________
Robert E.W. Fyffe, Ph.D.  
Vice President for Research and  
Dean of the Graduate School
ABSTRACT

Dias, Diogo. M.S.B.M.E, Department of Biomedical, Industrial and Human Factors Engineering, Wright State University, 2015. Design of a low-cost wireless NIRS system with embedded Linux and a smartphone interface.

Wireless near-infrared spectroscopy (NIRS) systems can help to reduce movement artifacts, and distraction due cables. Utilizing Embedded Linux (EL) can reduce size, development time, cost of a project and allow portability. The goal of this project is to develop a low-cost wireless small-sized NIRS system using EL and a Smartphone as the interface. This was achieved using a BeagleBone Black (BBB) with a deployed custom EL that: (1) controls two sources, (2) receives data from photodetectors (PDs), (3) processes, stores and transmits data via Bluetooth to an Android Smartphone. This device was implemented in under US$150 and its dimension was 9x5.5x4 cm. This is considerably cheaper and smaller than most of commercial NIRS systems. It can be applied in functional NIRS research and home care. More importantly our novel implementation of EL in NIRS will open up opportunities to further develop instrumentation using the modular nature of the Linux.
## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABAMAR</td>
<td>Accelerometer-based Motion Artifact Removal</td>
</tr>
<tr>
<td>APD</td>
<td>Avalanche Photodetector</td>
</tr>
<tr>
<td>app</td>
<td>(Smartphone) Application</td>
</tr>
<tr>
<td>ADT</td>
<td>Android Development Tools</td>
</tr>
<tr>
<td>BBB</td>
<td>Beaglebone Black</td>
</tr>
<tr>
<td>BSP</td>
<td>Board Support Package</td>
</tr>
<tr>
<td>CAPES</td>
<td>“Coordenação de Aperfeicoamento de Pessoal de Nível Superior”</td>
</tr>
<tr>
<td>CPU</td>
<td>Central Processing Unit</td>
</tr>
<tr>
<td>CW</td>
<td>Continuous Wave</td>
</tr>
<tr>
<td>Distros</td>
<td>(Linux) Distributions</td>
</tr>
<tr>
<td>DPF</td>
<td>Differential Pathlength Factor</td>
</tr>
<tr>
<td>ECG</td>
<td>Electrocardiography</td>
</tr>
<tr>
<td>EEG</td>
<td>Electroencephalography</td>
</tr>
<tr>
<td>EL</td>
<td>Embedded Linux</td>
</tr>
<tr>
<td>FD</td>
<td>Frequency Domain</td>
</tr>
<tr>
<td>fNIRS</td>
<td>Functional Near-Infrared Spectroscopy</td>
</tr>
<tr>
<td>FTP</td>
<td>File Transfer Protocol</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field-Programmable Gate Array</td>
</tr>
<tr>
<td>GND</td>
<td>Ground</td>
</tr>
<tr>
<td>GPIO</td>
<td>General-purpose Input/Output</td>
</tr>
<tr>
<td>GPL</td>
<td>General Public License</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>Hb</td>
<td>Deoxyhemoglobin</td>
</tr>
<tr>
<td>[Hb]</td>
<td>Deoxyhemoglobin Concentration</td>
</tr>
<tr>
<td>HbO₂</td>
<td>Oxyhemoglobin</td>
</tr>
<tr>
<td>[HbO₂]</td>
<td>Oxyhemoglobin Concentration</td>
</tr>
<tr>
<td>I/O</td>
<td>Input and Output</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>IIE</td>
<td>Institute of International Education</td>
</tr>
<tr>
<td>LD</td>
<td>Laser Diode</td>
</tr>
<tr>
<td>LED</td>
<td>Light Emitting Diode</td>
</tr>
<tr>
<td>LiPol</td>
<td>Lithium-Polymer</td>
</tr>
<tr>
<td>MBLL</td>
<td>Modified Beer Lambert Law</td>
</tr>
<tr>
<td>NFS</td>
<td>Network File System</td>
</tr>
<tr>
<td>NIR</td>
<td>Near-Infrared</td>
</tr>
<tr>
<td>NIRS</td>
<td>Near-Infrared Spectroscopy</td>
</tr>
<tr>
<td>OS</td>
<td>Operating system</td>
</tr>
<tr>
<td>OTA</td>
<td>Operational Transconductance Amplifier</td>
</tr>
<tr>
<td>PC</td>
<td>Personal Computer</td>
</tr>
<tr>
<td>PCB</td>
<td>Printed Circuit Board</td>
</tr>
<tr>
<td>PD</td>
<td>Photodetector</td>
</tr>
</tbody>
</table>
List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMT</td>
<td>Photo Multiplier Tube</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal-to-Noise Ratio</td>
</tr>
<tr>
<td>SSH</td>
<td>Secure Shell</td>
</tr>
<tr>
<td>TD</td>
<td>Time Domain</td>
</tr>
<tr>
<td>TIA</td>
<td>Transimpedance Amplifier</td>
</tr>
<tr>
<td>VM</td>
<td>Virtual Machine</td>
</tr>
<tr>
<td>WinCE</td>
<td>Windows CE</td>
</tr>
</tbody>
</table>
# Contents

1 Introduction .................................................. 1
   1.1 Purpose ................................................. 1
   1.2 What is a NIRS system? ................................. 1
   1.3 What about an embedded Linux? ....................... 2
   1.4 Significance ............................................ 3
       1.4.1 Why using a wireless device? .................... 3
       1.4.2 Movement artifacts ............................... 3
       1.4.3 Why using embedded Linux? ..................... 4
       1.4.4 The need of a low-cost device ................... 4
       1.4.5 The use of a mobile device on the design ....... 5
       1.4.6 New research fields and applications .......... 5
   1.5 Overview ............................................... 6

2 Near-Infrared Spectroscopy ................................... 7
   2.1 NIRS Historical Background .......................... 7
   2.2 NIRS Basis ............................................. 8
       2.2.1 Light Propagation in tissues ..................... 8
       2.2.2 Modified Beer-Lambert Law ....................... 10
       2.2.3 Kinds of NIRS .................................... 12
       2.2.4 NIRS treatments algorithms ..................... 15
       2.2.5 NIRS system Validation Methods ................ 16
   2.3 NIRS Limitations ...................................... 19
       2.3.1 Movement Artifacts ............................... 19
   2.4 NIRS Instrumentation .................................. 19
       2.4.1 Sources ........................................... 19
       2.4.2 Detectors .......................................... 21
       2.4.3 Source-Detector Positioning ..................... 22
       2.4.4 Optode configurations ............................ 23
       2.4.5 Optode housing ................................... 24
       2.4.6 Amplifiers ......................................... 24
       2.4.7 Power Supply ...................................... 25
       2.4.8 Processing Units ................................. 26
2.4.9 Wireless Options .................................................. 26
2.4.10 Wireless NIRS systems Examples .......................... 27

3 Embedded Linux .................................................. 30
  3.1 Embedded Linux Story ........................................... 30
  3.2 Embedded Systems .............................................. 31
    3.2.1 Alternatives to Embedded Linux ......................... 31
  3.3 Anatomy of Linux ............................................... 32
    3.3.1 Bootloader .................................................. 33
    3.3.2 Kernel ........................................................ 33
    3.3.3 Root Filesystem ............................................ 34
  3.4 Development considerations ................................. 34
    3.4.1 Components of a design environment and their Interaction .... 34
    3.4.2 General Public License .................................. 36
    3.4.3 Design Methodology ..................................... 36
    3.4.4 Relevant Concepts ....................................... 37
  3.5 Building tools .................................................. 38
    3.5.1 Popular embedded development boards .................. 38
    3.5.2 BBB overview .............................................. 40
    3.5.3 Buildroot ................................................... 43
    3.5.4 YoctoProject .............................................. 44
  3.6 Embedded Linux in Biomedical devices .................... 46

4 Methods .......................................................... 47
  4.1 Resources used .................................................. 47
    4.1.1 Choice of Materials ..................................... 49
    4.1.2 Software needed ......................................... 49
  4.2 Overview of the prototype ................................... 53
    4.2.1 Diagram .................................................... 53
    4.2.2 Development Plan ........................................ 54
    4.2.3 Chronogram ................................................ 55
  4.3 NIRS system Assembly ........................................ 56
    4.3.1 Nominal values .......................................... 56
    4.3.2 Circuits' Design ......................................... 57
    4.3.3 Circuits' Simulation ..................................... 57
    4.3.4 Testing in a Protoboard ................................ 58
    4.3.5 Creating a Cape ......................................... 60
    4.3.6 Setting the Battery ....................................... 61
    4.3.7 Recharging the LiPol Battery ............................ 63
    4.3.8 Rubber Case Design ..................................... 64
  4.4 Developing the Embedded Linux ............................... 66
    4.4.1 Configuring the Host computer ......................... 66
    4.4.2 Building Minimal System ................................ 68
    4.4.3 Deploying Minimal System ............................... 73
    4.4.4 Defining EL requirements ................................. 76
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Computer systems definition</td>
<td>2</td>
</tr>
<tr>
<td>2.1</td>
<td>NIR Absorption Range for Hb and HbO₂</td>
<td>9</td>
</tr>
<tr>
<td>2.2</td>
<td>Characteristic “banana shape” path of light propagating in tissue</td>
<td>10</td>
</tr>
<tr>
<td>2.3</td>
<td>Models of how light propagate in means</td>
<td>11</td>
</tr>
<tr>
<td>2.4</td>
<td>NIRS modalities examples</td>
<td>13</td>
</tr>
<tr>
<td>2.5</td>
<td>NIRS during arterial occlusion validation method</td>
<td>17</td>
</tr>
<tr>
<td>2.6</td>
<td>NIRS during brain hipoxia validation method</td>
<td>18</td>
</tr>
<tr>
<td>2.7</td>
<td>Optode configuration examples</td>
<td>23</td>
</tr>
<tr>
<td>3.1</td>
<td>Typical Linux architecture Components</td>
<td>32</td>
</tr>
<tr>
<td>3.2</td>
<td>Embedded Linux development setups</td>
<td>36</td>
</tr>
<tr>
<td>3.3</td>
<td>Popular development boards considered for the project</td>
<td>39</td>
</tr>
<tr>
<td>3.4</td>
<td>Main components of the BBB</td>
<td>41</td>
</tr>
<tr>
<td>3.5</td>
<td>Debug Serial Header connection position and cable</td>
<td>42</td>
</tr>
<tr>
<td>3.6</td>
<td>Buildroot Interface Example</td>
<td>43</td>
</tr>
<tr>
<td>4.1</td>
<td>PuTTY Terminal Emulator settings to connect to BBB via</td>
<td>50</td>
</tr>
<tr>
<td>4.2</td>
<td>Linux Ubuntu terminal</td>
<td>51</td>
</tr>
<tr>
<td>4.3</td>
<td>Diagram overview</td>
<td>54</td>
</tr>
<tr>
<td>4.4</td>
<td>Development Plan Diagram</td>
<td>55</td>
</tr>
<tr>
<td>4.5</td>
<td>Project Chronogram on a Gantt Chart</td>
<td>56</td>
</tr>
<tr>
<td>4.6</td>
<td>Sources circuit overview</td>
<td>57</td>
</tr>
<tr>
<td>4.7</td>
<td>Detector circuit overview</td>
<td>58</td>
</tr>
<tr>
<td>4.8</td>
<td>Simulation in the detectors circuit</td>
<td>59</td>
</tr>
<tr>
<td>4.9</td>
<td>Source-detector distance in this design</td>
<td>60</td>
</tr>
<tr>
<td>4.10</td>
<td>Circuit sketch of the designed cape</td>
<td>61</td>
</tr>
<tr>
<td>4.11</td>
<td>Designed BBB cape</td>
<td>62</td>
</tr>
<tr>
<td>4.12</td>
<td>Overview of adding the battery function on BBB</td>
<td>63</td>
</tr>
<tr>
<td>4.13</td>
<td>Recharging battery setup</td>
<td>64</td>
</tr>
<tr>
<td>4.14</td>
<td>Cut neopropene sheets used on the case design</td>
<td>65</td>
</tr>
<tr>
<td>4.15</td>
<td>Prototype in the neopropene case</td>
<td>66</td>
</tr>
<tr>
<td>4.16</td>
<td>Hob machine and image selection screen</td>
<td>68</td>
</tr>
</tbody>
</table>
List of Tables

2.1 Brief comparison of NIRS systems modalities .......................... 12
2.2 Comparison between LD and LED for NIRS systems .................. 20
2.3 Qualitative Comparison between PDs for NIRS systems ................. 22
2.4 Commercial NIRS systems characteristics ............................... 27
2.5 Literature NIRS systems characteristics ................................. 29
3.1 Features Comparison between BBB and RPi ............................... 39
4.1 Items list to develop the system ........................................... 47
4.2 Development items and tools used in this project ....................... 48
4.3 Specifications of the NIR LEDs used ..................................... 49
4.4 Voltages on the BBB’s I/Os ................................................. 56
4.5 Measured voltages on the circuit on a protoboard in a well-lit ambient situation .......................................................... 60
4.6 BBB battery pins .............................................................. 62
4.7 Custom EL package list ....................................................... 77
4.8 Logfile example of the system offline mode ............................... 84
4.9 Distances between sources and detectors ................................ 88
4.10 Extinction coefficients used ................................................. 89
4.11 Differential Path Length Factors Used ................................... 89
5.1 Mechanical specifications of the design with and without the rubber case . 99
5.2 Battery specifications of the design [99]. ................................ 100
Acknowledgment

I would like to take this opportunity to extend my thanks firstly to my family and my girlfriend for their daily support during my studies (even 5,000 miles away).

I would like to thank also my sponsors “Coordenação de Aperfeiçoamento de Pessoal de Nível Superior” (CAPES) and “Institute of International Education” (IIE) for granting me a scholarship through the science without boarders program.

One important thanks goes to Wright State University, specially to the BIE Department for helping me a lot performing my research due its structure, for buying important research items used in this project and also for helping me to present this work on my first conference as a poster presenter at BMES annual meeting 2015 at Tampa.

Other special thanks goes to the all the Professors I could get some advice from and for those that lectured me during my graduate studies: Dr. Thomas Hangartner, Dr. Julie Skipper, Dr. Frank Ciarallo, Dr. David Reynolds, Dr. Nasser Kashou, Dr. Jack Jean, Mr. Dave Kender, and Dr. Subhashini Ganapathy. I would like to stand out Dr. Jack Jean in this group of professors for providing me some header pins that helped me a lot to design a cape to the BBB, what was essential to the success of this project.

I would like to stand out my gratitude to the Biomedical Imaging Lab’s that helped a lot by sharing knowledge, research experiences and friendship. My thanks go specially to Irfaan and Emma for sharing their experience with NIRS Experience that was essential during designing my prototype. I would like also to thank Maddie Jones for helping me to review a manuscript that hopefully will result the work performed in this thesis to be accepted into a journal.

I wish to thank all the friends and colleagues I have made so far from all around the world during almost 2 years I have been living here in Dayton, some of the most remarkable ones are Andre, Marques, Zorzi, Miriã, Renata, Lays, Luis, Seung, Martin, Dave, Taylor, MK, Samantha, Jared, Lucile, Kelsey, Chase, Alex, Tyler, Sandy, Jan, Ramzy, Rafa, Abdullah, Sami, Jameel, Jimmy, Estelle, Livia, Lilia, Jeffrey, Prateek, Chirag, Ali among many
others. They helped me a lot to feel like home and to learn different cultures from around the world, in this first experience I have been abroad.

I would like to extent my thanks to some friends back in Brazil that helped me to grow love to EL that is something I simply believe to be potentially helpful on the development of alternative technologies. I would like to thank my friends that I had my undergraduate in Biomedical engineering that keep sharing their experiences in this area until today two years later we graduated, and my special thanks in this group goes to Guilherme Marques that helped me a lot on some tips to solder my battery pins and the cape I use on the the prototype that will be presented, and Felipe Lisboa for proofreading my thesis and for giving me some tips to improve its writing.

I am particularly grateful for the Committee members Dr. Jack Jean and Dr. Ulas Sunar for accepting the invitation to be part of this extremely important moment in my career and life.

Lastly I would like to express my deep gratitude to my advisor Dr. Nasser Kashou, firstly for helping me to develop the idea of this thesis, then I would like to thank him also for all the advice and guidance he has given me throughout this year, the encouragement he gave me and the other students to publish our work and present our work at important conferences, particularly this project that we could present in two conferences and we expect to get our work published soon. I would like to thank for all the knowledge he shared during his lectures and during Biomedical Imaging Lab's Group meeting, and individual meetings and finally I would like to thank him for his strong belief in his students' work, in the university he works and in science... and this makes all the difference on inspiring all of us to make outstanding work, and I'm really proud and grateful for choosing him to be my advisor.
Dedicated to

my beloved angry girlfriend and my parents for all the support
Chapter 1 - Introduction

This section will point out the reasons and motivations to this thesis be written and the importance of this work. Additionally, this section will have an overview about the following chapters.

1.1 Purpose

This thesis intends as a primary goal to develop a wireless NIRS system using EL, and as a secondary goal the design will be focused to reduce the device’s size, weight, cost and provide a smartphone as the interface.

1.2 What is a NIRS system?

Near-infrared spectroscopy (NIRS) is a non-invasive and non-ionizing optical method that can measure physiological parameters such as blood volume and tissue oxygenation changes [1, 2]. In the near infrared (NIR) range (wavelengths between 650-900nm) [3] the interaction between NIR with water, lipids and other chromophores is relatively low compared to deoxyhemoglobin (Hb) and oxyhemoglobin (HbO₂) that are present in blood, and there is also a contrast between Hb and HbO₂ interaction in this same range, allowing to measure backscattered NIR photons with suitable detectors and relate these photons to the concentration of Hb and HbO₂ ([Hb] and [HbO₂]) [3, 4, 5] in an irradiated region with
NIR sources. Further information is provided in Chapter 2 talking about NIRS in more details.

### 1.3 What about an embedded Linux?

Before defining what an EL is, the definitions of embedded system, and computer are needed. A computer according to a Von Neumann's machine architecture (fig. 1.1(a)) and a Harvard architecture (fig. 1.1(b)) defines it as the combination of a central processing unit (CPU) with a memory, differing that while the first one the memory reads and writes in the same place, in the second one there is one input for reading and one output for writing [6].

![Computer systems definition. (a) Refers to the Von Neumann's machine, where there is a Memory that both reads and writes to the CPU and (b) refers to the Harvard architecture, where there is a memory to read and other to write. Adapted from Wolf [6]](image)

An embedded system is any device that includes a CPU and a memory and is not intended to be a general purpose one like a Personal Computer (PC) or laptops. Other common definition is that an embedded system is a computer that does not look like one in a first look, in a user’s perspective. Examples of embedded systems range from fax machines, automobiles to clocks and all of these examples use microprocessors. Making embedded systems is an important skill for device designers that want to implement a solution to a specific task. It is also a good option when resources such as battery, memory, process power are limited on a project [6, 7, 8, 9, 10]. Lombardo gives another good example of
what an embedded system is: a general purpose computer is like a Swiss knife, while an embedded system is a simple knife, and if one just want to cut vegetables the knife does all it takes for the job [11]. Regarding EL, it can be seen as an embedded system that uses a Linux Kernel in the operating system (OS) [10, 12, 13]. Further information is provided in Chapter 3 talking about EL in more details.

1.4 Significance

1.4.1 Why using a wireless device?

According to Jovanov et al. [14] wires in medical equipment can limit the movement of the patient and attract his/her attention during continuous monitoring. Wires also come up with tripping hazard possibilities in hospital environments as said by Golmie et al [15]. Muehlemann et al. [16] add that in commercial NIRS systems, which use electrical and optical cables, there is the possibility that NIRS systems cabling moves the light sensors and consequently worse the coupling of the optodes with the tissue, resulting in disturbance due to movement artifacts. They also affirm that cabling increases the size and the weight of the equipment, complicating the device’s handling, and increasing the equipment’s price mainly if optical fibers are used.

1.4.2 Movement artifacts

As said on the previous subsection the movement of the sensors and sources and consequently worsen the coupling of the optodes to the skin causing movement artifacts [16], the removal of such artifacts in NIRS systems is usually made by algorithms [2, 17, 18], if the result of the removal of this artifacts is satisfactory there are more possibilities for studies in NIRS field with free movement subjects like athletes’ muscles in movement, subjects during social interaction [16], and cases where one cannot, or the control of subjects is hard
'like sleep analysis [18], neonates [17] and animal studies [16, 17].

1.4.3 Why using embedded Linux?

The use of an EL is advantageous when one think about development time and cost (if one uses an open-source version) because there are available free reusable drivers and applications, besides the Linux community support [8, 19]. Another major feature of Linux is the kernel structure and the possibility to develop loadable modules for additional devices/sensors easily [20], like the use of a strain gauge in order to use the acceleration data in a movement artifact removal algorithm as done by [2]. NIRS commercial systems are inflexible according to their application [21] and the modular feature of Linux could be helpful in developing a more flexible device.

An embedded project has a potential to reduce size of the equipment, price, and make obsolete the use of a whole PC together with the system, because EL can optimize the use of resources of a system [10, 22]. There are already similar applications using embedded devices like the NIRS system using Field-Programmable Gate Array (FPGA) sensors developed by Safaie et al. [21] but they had the only option to interface with a PC, one using Windows CE and an ARM9 microcontroller for processing, storing and displaying Data developed by Zhang et al. [22], and the other one using an EL (uCLinux) but the system developed by Haensse et al [23] was not wireless. One important factor of this study is that this could be the first wireless NIRS system based on a EL that is able to show real-time data on an Android phone.

1.4.4 The need of a low-cost device

The need of low-cost biomedical devices are growing and an affordable one could help places poorly served by medical technology, remote places and low-cost NIRS devices also come up with the need of more affordability and possibility of using NIRS in homecare
applications in order to reduce healthcare courses [22].

1.4.5 The use of a mobile device on the design

There is an increasing use of smartphones by everyone, including health care professionals, the growing popularity of smartphones can be explained by smartphones’ portability, usability and rapid-access to information. Showing the big picture, the number of smartphones owned by the population who is older than 13 years old, in the United States, is about 51% of this population (what it was about 113 million units in 2012), and worldwide 1.5 billion cellphones are estimated, and it is estimated that: the number of smartphones will be up to 6.5 billion devices in 2018 [24]. The increasing number of Smartphone applications (apps) applied to medicine to monitor, data sharing, calculations, among other applications [25, 26] made also the FDA to come up with standards and guidance to mobile applications in the year of 2013 [24]. There are some advantages of using a smartphone as a healthcare support device, because the professionals have rapid access to data, smartphones usually have relatively good resolution displays for monitoring purposes, and it usually have apps with satisfactory usability [24, 25].

If one’s looked for in literature this is the first wireless NIRS system using EL with a smartphone as the Graphical User Interface (GUI).

1.4.6 New research fields and applications

An easily transportable NIRS system provide new fields of application like emergency medicine, [16], a wireless functional near-infrared spectroscopy (fNIRS) wearable device also offers new possibilities of following up the gait of a patient with impaired movements, following up therapeutic treatments on stroke victims and applications in sports science [27]. Some studies also says that wireless NIRS can be used for Brain-machine interface, a wireless system would also be useful for the development of further applications using
1.5 Overview

In the next chapter (chapter 2), NIRS's theory, concepts, instrumentation and some NIRS devices examples in literature and commercial products are going to be talked about.

Chapter 3 contain the story, concepts, the anatomy of an EL, some building tools and examples of biomedical devices that used EL in literature.

In chapter 4 the materials and methods used in this project are showed, besides the development plan and chronogram planned to this project. This development plan is basically divided in four stages of development “Items used”, “Hardware development”, “Software Development”, “GUI development”, and “Validation”.

In chapter 5 the device's technical specifications of the system are inserted. The resulting validation of the device, the specifications, and some development comments are also showed.

Chapter 6 consists on a discussion about the achieved results and some development issues throughout the project and how those issues were faced in order to bring out the final prototype. A conclusion and future plans with this device are also part of this section.

In the end of the thesis the references are showed using the IEEE - Transactions in Medical Imaging citing style, and the Appendix explain the structure where the codes used in this project are in a github repository.

A pattern is used to indicate command lines codes inputted in a Linux terminal through black background and green font as showed in this example below:

```
example
```

Another pattern used in this thesis is the pink text that works as hyperlinks in the digital copy of this thesis, for example this link that takes to the website of Biomedical Imaging Lab at Wright State, where most of the work of this thesis was done.
Chapter 2 - Near-Infrared Spectroscopy

2.1 NIRS Historical Background

Hippocrates (460 —370 BC) was one of the first documented person to relate light to medicine, observing that exposure to sun light had beneficial effects. Two millennia later, we are still trying to understand how light interacts with biological tissues [29]. Other important event was the creation of spectrophotometry and colorimetry made by chemical analysis from Gustav Kirchoff (1824-1887) and Bunsen (1811-1899), that analyzed that certain chemicals absorb light in proportion to the chemicals concentrations. Contributors like Tyndall in 1873, were extremely important to propose dual-wavelength spectrophotometry and improve the measurements possibilities, one of them is the use for optical clinical monitoring [29]. The first tries were with visible light and date back from the early nineteenth century, when physicians started to look closely on how the light could help the professionals in diagnostic, like Cutler in 1929, Bright in 1831 and Culing in 1856 used light to illuminate the breast and detect large tumors [27, 29]. In 1876, Von Vierodt analyzed the light penetration in tissue is changed when blood circulation was occluded [27].

NIRS for in vivo applications starts with oximetry field, which is an invasive or non-invasive blood oxygenation technique, Hübner in 1900 observed that the extinction of light by Hb was made by two wavelengths, and this could allow the assessment of the degree of oxygenation in tissue. The first non-invasive oximetry clinical application to observe oxy-
generation in the ear was made by Matthes in 1935, two years later Millikan developed Tyn-
dall's spectrophotometer to observe Hb and myoglobin in a cat. Four years later Millikan
also adapted his approach for oxygen saturation monitoring in humans [29]. The Beer-
Lambert law is promptly attributed to Lambert in 1760, however Bouguer was the first to
come-up with this law in 1729. Horecker, in 1943, proposed to measure optical-absorption
spectra of Hb and HbO$_2$ using Beer-Lambert law to solutions with Hb derivatives, however
the early oximeters could not correct for the many compounds found in real tissue. In 1949,
Wood and Geraci tried to overcome the problem by correcting the measurements of an ear
subtracted by the measurements of a bloodless ear [29]. As the original Beer-Lambert Law
does not take into account scattering a recently modified version was proposed in 1988 by
Delpy. Other improvements made by the diffusion equation to describe light transport by
tissue were made by Arridge et al, in 1992, and Patterson et al, in 1989 [27]. Hewlett-
Packard came with the first blood oximeter with 8 wavelengths to derive oxygen saturation
at 1970, right after that in 1974 Aoyagi et al came with a pulse oximeter proposal that later
would be commercialized by Minolta [29].

Butler and Norris, in 1960, examined that NIR light could be used to examine plant
tissues and the human hand and Jöbsis [30] came with the use of NIR light to record brain
and myocardial oxygen concentration, and he is considered the creator of the NIRS for in
vivo applications, after his discoveries several NIRS systems were designed, and a large
range of applications were tested on the last 40 years [27, 29, 31].

2.2 NIRS Basis

2.2.1 Light Propagation in tissues

When light is applied to biological tissues, light can either be reflected, absorbed,
transmitted or scattered by chromophores, that are the part of the molecules that are re-
Figure 2.1: NIR Absorption range applied to biological tissue, where Hb and HbO₂ curves are indicated. The isobestic point refers to the wavelength that the absorbance between these two chromophores are about the same. Modified from Rolfe [29]

lated to the molecule’s color. The reflectance is more dependent on the angle which the light is applied, while the scattering and absorption are dependent on the wavelength and the tissue optical properties. In the NIR range 700-1300nm a larger amount of light can be transmitted through longer distances in biological fields (around 0.5-3 centimeters deep [32], while above 1300nm the water absorbs most of the light and below 700nm the scatter is the predominant effect. Blood oxygenation information could be achieved through successful NIR transmission in tissue, where the HbO₂ has a greater attenuation at a wavelength around 750nm, while the Hb is around 830nm, as seen in fig. 2.1. Two or more wavelengths are used in order to make more accurate measurements, because when one is analyzing more than one chromophore more information is needed [30, 31, 33, 34].

The light has a random pattern of attenuation when applied to a tissue, when running Monte Carlo simulations, the most probable position in order to detect light is the one positioned the detector some centimeters away from the source and the light has this characteristic “banana shape” pattern [3, 37], as seen in fig. 2.2.
2.2.2 Modified Beer-Lambert Law

A light beam going through a mean and the light attenuation can be described by the Beer-Lambert law (eq. 2.1).

\[ \ln\left(\frac{I_0}{I}\right) = \epsilon CL \]  

(2.1)

Where \( I_0 \) is the light before going through the mean, \( I \) is the light after going through the mean, \( \epsilon \) is the extinction coefficient, \( C \) is the concentration of a chromophore, and \( L \) is the optical path length [3], a simple model is showed in fig. 2.3 (a).

However the description of light propagation in tissue is more complex than the Beer-Lambert Law model, because tissue is not homogeneous like a simple mean [36]. One of the light paths that light has when propagated in living tissue is a “banana shape” one as previewed by Monte Carlo Simulations [3, 37, 34] illustrated on fig. 2.3 (b), the other path that light can have is scattering due refraction index variation between cells and their constituents, which causes only a small fraction of light to be detected, making the process
Figure 2.3: Models of how Light propagate in means, (a) shows the light propagating in a simple mean usually used in a Beer-Lambert Law example, (b) shows light propagating in a biological mean, an example usually used to describe how the Modified Beer Lambert Law (MBLL) works. It also shows a SRS system, where there are detectors placed with different distance from each other. Modified from Mansouri and Kashou [3] to acquire data harder to quantify the total light loss and the absorbers in the tissue [38].

The Beer-Lambert Law to relate light intensity variation to a chromophore concentration formula, the equation 2.2 is showed below.

\[
\Delta OD = \ln\left(\frac{I_0}{I}\right) = \epsilon \Delta CLB
\]  

(2.2)

Where \(\Delta OD\) is the difference between the final and initial optical density, B is the differential path length factor (DPF) and \(\Delta C\) is the change in the chromophore concentration.

In order to quantify more accurately the variation in concentration of two chromophores, two wavelengths or more are needed. The most common application is to use the contrast of Hb to HbO\(_2\) with modified Beer-Lambert Law (MBLL), because these hemoglobins are the main absorbers of NIR light in the NIR range. For one wavelength is used using equation 2.3 [3, 34].

\[
\Delta [OD] = (\epsilon_{HbO_2}^\lambda \Delta [HbO_2]^\lambda + \epsilon_{Hb}^\lambda \Delta [Hb]^\lambda) B^\lambda L
\]  

(2.3)
Where $\lambda$ represents a specific wavelength. When we use two wavelengths the MBLL in order to calculate the concentration of Hb ($\Delta[Hb]$) and HbO$_2$ ($\Delta[HbO]$) we have the following two equations (2.4 and 2.5) [3]:

$$\Delta[Hb] = \frac{[\epsilon_{HbO}^2(\Delta OD_{\lambda_2}) - \epsilon_{HbO}^1(\Delta OD_{\lambda_1})]}{[\epsilon_{Hb}^1 \epsilon_{HbO}^2 - \epsilon_{Hb}^2 \epsilon_{HbO}^1]L]}$$

(2.4)

$$\Delta[HbO_2] = \frac{[\epsilon_{HbO_2}^1(\Delta OD_{\lambda_2}) - \epsilon_{HbO_2}^2(\Delta OD_{\lambda_1})]}{[\epsilon_{Hb}^1 \epsilon_{HbO_2}^2 - \epsilon_{Hb}^2 \epsilon_{HbO_2}^1]L]}$$

(2.5)

Where $\lambda_1$ and $\lambda_2$ represent different wavelengths. Other studies like Kim et al [2], used the same formula to calculate changes in [Hb] and [HbO$_2$].

### 2.2.3 Kinds of NIRS

There are 3 kinds of NIRS systems time domain (TD), continuous wave (CW), and frequency domain (FD) developed so far, each one has their own advantages and disadvantages. The illustration of each kind of NIRS is showed in fig. 2.4. Table 2.1 summarize each kind of NIRS system’s pros and cons. There is also the SRS systems that are going to be explained further.

<table>
<thead>
<tr>
<th>Method</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>CW</td>
<td>-Low cost</td>
<td>-Cannot differentiate scatter and absorption</td>
</tr>
<tr>
<td></td>
<td>-Low power consumption</td>
<td>-Cannot go deep in tissue</td>
</tr>
<tr>
<td></td>
<td>-It Can measure [Hb] and [HbO$_2$]</td>
<td></td>
</tr>
<tr>
<td>FD</td>
<td>-High temporal resolution</td>
<td>-High noise due to scatter effects</td>
</tr>
<tr>
<td></td>
<td>-Sampling rate</td>
<td>-Needs more development</td>
</tr>
<tr>
<td></td>
<td>-Info about scattering/absorption</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-Cheaper than TD</td>
<td></td>
</tr>
<tr>
<td>TD</td>
<td>-High spatial res.</td>
<td>-Large dimension,</td>
</tr>
<tr>
<td></td>
<td>-Deeper penetration depth</td>
<td>-Large cost,</td>
</tr>
<tr>
<td></td>
<td>-Info about scattering/absorption/time-of-flight</td>
<td>-Low temporal resolution</td>
</tr>
</tbody>
</table>
Figure 2.4: NIRS modalities examples, (a) CW NIRS, where the incident light is simply attenuated by the mean (b) FD NIRS, where the incident sinusoidal light is attenuated and dephased by the mean (c) TD NIRS, where the incident ultra-fast pulses of light, are attenuated and used to get a temporal point spread function. Modified from Scholkmann [27]

**CW NIRS**

The CW NIRS systems (fig. 2.4 (a)) are the most common developed and the most inexpensive technique, this kind of NIRS system consists on applying a constant or a low-frequency modulated light source into tissue and then detecting the attenuated signal. The CW NIRS systems’ advantages are the price, less power consumption, and the possibility to measure [Hb] and [HbO₂]. These systems’ disadvantage is the limitation on differing scattering and absorption effects on the light attenuation in biological tissue, it has also the limitation for not being able to evaluate the tissue in deeper areas, because most of the light is attenuated on the first centimeters into the tissue [3, 27].
FD NIRS

The FD NIRS systems (fig. 2.4 (b) ) use a continuous amplitude modulated light with a frequency around 10-100 megahertz applied into tissue and the resulting signal is a phase shifted one with its amplitude reduced. FD NIRS systems’ good points are a fast sampling rate, relatively temporal resolution, the ability to gather information about scattering and absorption and FD systems are cheaper than TD systems. These systems disadvantages are the scattering noise, and the lack of further development [3, 29, 39].

TD NIRS

The TD NIRS technique (fig. 2.4 (c)) consists on using a fast light pulse into tissue (picoseconds'step) in order to get a temporal point spread function resulting from the output of the pulse after being scattered by the tissue. TD systems can measure the time of flight of the photon, and consequently gather information about scattering and absorption, lastly it can provide a deeper penetration of photons [3, 27, 39].

The bad aspect of the TD systems is that they require ultra-fast lasers able to perform the pulses at a high frequency, making the TD systems expensive and bulky. They also have lower temporal resolution than the other techniques. However they have the best spatial resolution among the NIRS systems [3].

SRS NIRS

The SRS is the technique that uses a light source and a series of detectors in different or adjacent spots away from the source and then measurements in these several points could be made to gather more information from reflected photons [29, 40]. SRS could be used with several configurations [29], and one example of this kind of NIRS is showed on fig. 2.3 (b) [29].
2.2.4 NIRS treatment algorithms

Movement Artifacts Removal

Kim et al used an accelerometer and an active noise cancellation algorithm that subtracted the motion artifacts of the corrupted signal by the estimated movement of the accelerometer’s measurement in order to filter the NIRS signal. The error power of the signal was evaluated with an adaptative filter to minimize signal error. The group could effectively remove head motion artifacts from the signals, however this group algorithm added a 0.5s delay on comparison to the real-time signal [2].

Virtanen et al proposed another algorithm for motion artifact removal that used an accelerometer, called as accelerometer-based motion artifact removal (ABAMAR), this algorithm consists in aligning the sensors with the tissues with the NIRS optodes, in this study the optodes were placed at the head of the subject and the measurements with the subject at rest were defined as baseline. After a moving event was detected with an interval of 20 seconds of a previous motion, and this event was at least 1 second long this event is considered as a movement artifact, and it was removed, these thresholding times to define movement artifacts were defined through visual inspection [18]. Virtanen et al's algorithm also proposed a compensation to baseline shifts that were defined as the mean amplitude of signals after 15s of recorded data, after each movement artifact is detected the baseline shifts were corrected [18].

Scholkmann et al's group’s algorithm also came up for movement artifact's removal and they called it motion artifact reduction algorithm. This algorithm consisted on calculating the moving standard deviation of the discrete series of the signal, detecting the start and end point of the movement artifact, segmenting the signal on places with and without movement artifacts, creating and subtracting the spline interpolation on each segment and then reconstructing the signal. They achieved considerable reduction of artifacts and increased the signal quality [17].
Baseline Algorithm

This algorithm consists on keeping the data in the same base regardless the background light. It consists in three stages: first collecting each detector signal with the sources off, then the measured values are subtracted by the following measurement values with the sources on. Lastly the sources are applied in a pulsed pattern during a period and an array during this period is stored, the values of this array are averaged and then the code automatically subtract this average from the measured values during the analysis period [22, 36].

Averaging Signal

Averaging signal consists simply on summing up the signal and then dividing this sum by the number of samples of a part of the signal in order to improve the Signal-to-Noise ratio (SNR) of the acquired data. This technique considers that the noise is random, uncorrelated with the signal and has a mean of 0. This technique reduces the noise of a signal as the number of samples is increased. The Averaging Signal Technique is commonly used on biomedical signals such as Electrocardiography (ECG), oximetry and Electroencephalography (EEG) [41, 42].

2.2.5 NIRS Validation Methods

Arterial Occlusion

In this validation test, the optode is placed on the left forearm, and a pressure cuff is fastened on the upper forearm in order to induce arterial occlusion. Then a 30-second baseline period (without the cuff fastened) is obtained and a 90-second period with the cuff applying a pressure of 260mmHg is recorded, after that the pressure is reduced back to normal and then the blood flow goes back to normal. A pattern in fig. 2.5 is expected to be
seen, when the arm is occluded the [Hb] tends to go up while the [HbO$_2$] goes down, and when the pressure is reduced the opposite happens [2]. Other prototypes used an analogue validation method [21].

![Figure 2.5: NIRS during arterial occlusion validation method and the NIRS expected signal, the red line represents the [HbO$_2$] and the blue one is the [Hb], the occlusion moments are indicated by the colored areas. From [2]](image)

**Brain hipoxia**

For this validation method the optodes are placed on the forehead of the subject and then 45 seconds of the signal on a baseline state is recorded. After that the subject is required to hold his/her breathe for the longest time he/she can take and this signal is recorded, after that the subject can breath normally for 90 seconds. The expected signal for this part is showed on fig. 2.6, after some time not breathing the [Hb] tends to go up while the [HbO$_2$] goes down, and when the subject catch his breathe back the opposite happens [2]. Other prototypes used an analogue validation method [21].
Figure 2.6: Brain hipoxia validation method and its expected signal, the red line represents the $[\text{HbO}_2]$ and the blue one is the $[\text{Hb}]$, the breath holding moment is indicated by the colored areas. From [2]

**Finger Tapping**

Muehlemann et al’s group commented that they have used a finger tapping experiment to check the validity of their NIRS system prototype. The experiment is realized by placing the sensors on the top of the motor cortex, and then the subject stayed 20 seconds resting and other 20 seconds tapping his/her right thumb and right index finger. Muehlemann et al’s group expected NIRS signal is the one that when the optodes are placed on the head the $[\text{HbO}_2]$ should have a slightly higher concentration than the $[\text{Hb}]$ during the finger tapping. [16]

**Going up and down Stairs**

Safaie et al’s group used a different protocol where the group has used two of their systems one positioned at the forehead and the other at the largest posterior muscle at the calf. The experiment was run with a 60-second rest for the subject and then the subject ascended 10 stair steps, after another rest period the subject descended the ten stair steps.
They could observe a reduction in both [Hb] and [HbO₂] during exercise in the muscle optodes [21].

**Phantom tissue/liquid**

Some studies used a silicon model to mimic biological tissue, and others used a liquid container to evaluate the sensitivity map of the system [21, 36, 43].

### 2.3 NIRS Limitations

#### 2.3.1 Movement Artifacts

NIRS signals are not directly affected by movement itself, but the movement affects the contact of the sensor with the tissue [18], so in order to have an optimal signal for the NIRS system, the sensors must be steady and coupled to the subjects skin, some factors harm these optimal design due cables moving the sensors [16] and due the non-flat structure of our body. A suggestion to get rid of these problems is the use of wireless systems for not having cables pulling the sensor from the coupling, the use of flexible Printed circuit Board (PCB) to position the sensors and sources in a perfect way [2, 16] and by using of motion artifacts removal algorithms [2] (discussed on section 2.2.4). These artifacts also limit freely-moving subjects like athletes during exercises [16].

### 2.4 NIRS Instrumentation

#### 2.4.1 Sources

There are two options that are mostly used as sources for NIRS systems according to Safaie et al’s group. The use of the Light Emitting Diode (LED) is more advantageous
because LEDs’ low-cost, low power consumption, easy circuitry, usually no harmful to the eyes and by using LEDs make the fiber optics use obsolete. These characteristics are good for a wireless device. The group have also pointed out the limitations of heating and the lack of possibility of using the Laser Diodes (LDs) direct to the skin [21]. However Rolfe points out that there are limitations on LEDs’ power and consequently the penetration in the skin, The LED’s spectral bandwidth is 10 times bigger than the LDs. A technical overview of both source types is showed on the table 2.2 [29].

Table 2.2: Comparison between LD and LED for NIRS systems

<table>
<thead>
<tr>
<th>Feature</th>
<th>LED</th>
<th>LD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Consumption</td>
<td>2-10mW</td>
<td>10W</td>
</tr>
<tr>
<td>Fiber Optics Cable</td>
<td>not needed</td>
<td>needed</td>
</tr>
<tr>
<td>Cost</td>
<td>very low</td>
<td>low</td>
</tr>
<tr>
<td>Safety</td>
<td>Usually not harmful</td>
<td>Temperature increase issue</td>
</tr>
<tr>
<td>Spectral Bandwidth</td>
<td>50nm-long</td>
<td>5nm-long</td>
</tr>
<tr>
<td>Penetration into tissue</td>
<td>low</td>
<td>high</td>
</tr>
<tr>
<td>Electronics needed</td>
<td>Simple/Small</td>
<td>Complex/Bulk</td>
</tr>
</tbody>
</table>

Rolfe comments also that the LDs provide a narrower spectral bandwidth and the possibility of pulsatile operation at a really fast frequency (<500ps pulses), LDs’ disadvantages are that it takes some time to heat up the LDs to achieve their stability (up to 1h for some LDs) [29, 44].

Pulsed solid state lasers are other possible option to have as the sources, being able to produce pulses even faster than the LDs 100 ps - 100fs, the pulsed solid state lasers are able to work with 100Mhz repetition rates, however they are too bulky to use in clinical practice and it takes almost 10s to change from frequency to frequency, they are also more expensive when compared to LEDs and LDs [27, 44].

Safaie et al’s NIRS system emitters are controllable and they can be either pulsed or continuous, CW is cheaper and simpler [21], Achigui et al’s system has 3 emitters and an activation pulse for each emitter one at a time with a varying frequency of 10-50khz, Achigui et al have also used a CW configuration and have said that this configuration allows
a light-weighted and simple monitoring system [32]. Bozkurt et al’s system suggested an adjustable LED driver for people with darker skin, because melanin tends to attenuate NIR light as well, so with a higher intensity light can be compensated this photons loss [34].

Another important feature to be considered is to choose the most suitable wavelengths in order to have the minor crosstalk effect possible, and the higher contrast between [Hb] and [HbO₂] [45, 46] possible. Strangmann et al found that when using both wavelengths above 780nm, a high cross talk and low separability is obtained, while the opposite happens when using a wavelength below 720nm and one above 730nm [45], some suggested combinations showed better focal change during oxygenation measurements were 760nm or 690nm paired with 830nm [46].

2.4.2 Detectors

According to Rolfe the Photo Multiplier Tube (PMT) is a good option when one needs a lot of sensitivity, but it needs a high-voltage source, cooling and protection from high-intensity light [29], it also is too big, heavy and expensive [21].

For Scholkmann et al, the Avalanche Photodiode (APD) is the best option for the receptor in a NIRS system, considering this detector’s size and the considerable sensitivity [27]. Safaie et al considered that the silicon photo diodes (SPDs) is advantageous by providing enough sensitivity without having bias due to high voltage that happens in PMT and APD. Achigi et al also used regular PDs [21, 32].

Photodiode OPT101 (Texas instrument) is good because there is no need for using circuitry like ITA since an ITA has already one on-chip in the OPT101, the detector is also relatively small (it is a 0.4x0.4 inch chip) [22]. A summary comparing the cited detectors is showed on table 2.3.
Table 2.3: Qualitative Comparison between PDs for NIRS systems

<table>
<thead>
<tr>
<th>Feature</th>
<th>PD</th>
<th>APD</th>
<th>PMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Consumption</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Use of fiber cables</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Cooling needed</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>High-Voltage source</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Size</td>
<td>Small</td>
<td>Medium</td>
<td>Big</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>Small</td>
<td>Medium</td>
<td>High</td>
</tr>
</tbody>
</table>

2.4.3 Source-Detector Positioning

For the NIRS signal, a very important factor is that the source-detector coupling should be constant, otherwise the light emission and detection are affected and the signal gets noisy, this happens because the sources and detectors can be moved during analysis due some instability of the source-detector coupling, inclination of the sources and detectors could also result in movement artifacts on the NIRS signal. Through the use of caps (in brain analysis) or other housing types making the sources and detectors to be more stable during the analysis is possible [47]. The distance between source and detector is directly proportional to the depth that lights goes into tissue, and that is another reason why it is important in NIRS [47, 48]. The distance between the source and detectors, and the optodes’ configuration seems to be controversial, because there are many applications and possibilities of configurations. Kassab [47] cites that the source-detector distance is usually 3±0.5cm,

Mansouri and Kashou [3] state that by varying the distance between the sources and detectors the penetration of the light on the region can be controlled, Mansouri and Kashou also has showed distance differences of 4.8cm to 2cm as the optimal range that detectors and sources should be apart. Patil et al [43] use their design with 2,3 and 4cm distance between sources and detectors and Patil et al showed that for closer distances the light intensity suffered quicker decline in intensity, and Patil et al confirmed that the greater the distance the greater was the penetration. Strangmann et al [39] cites the importance
of considering the power applied in tissue and says that typically the photons detected at 4cm away from the source are 10pW photons and that the reduction factor of photon power magnitude are around 7-9 times. Scholkmann et al [27] shows that the commercial NIRS systems ranges from 1.5-4cm of source-detector separation and mostly of NIRS systems are adjustable making the analysis at larger or shorter distances possible.

Besides the detector-source distance, the angulation of the optodes are also important to consider for small distances between the source and detector [48].

### 2.4.4 Optode configurations

Mansouri and Kashou [3] comments about two optode configurations: the first one is a circular one to round tissues such as the breast analysis that surrounds the tissue with sources and detectors with the idea of using for DOT imaging. The second one is by using
two thin parallel glasses to compress tissue in order to reduce the tissue's light attenuation, but the compressed tissue make the tissue optical properties change and harder to get depth resolution.

Strangmann et al [39] provides two examples of detectors and sources configurations on the fig. 2.7, the first one on fig. 2.7(a) represents a point measurement that consists measuring two dominant absorbers and then comparing the absorbers’ concentration this is an example of NIRS, The second one showed on fig. 2.7(b) shows an imaging instrument that requires numerous sources and detectors that provides overlapping measurements analyzing the sensitivity through image reconstruction, this configuration can provide DOT measurements.

2.4.5 Optode housing

Some prototypes suggested the use of flexible PCBs for the sources and detectors in order to have a better fit since most of human tissue are curved surfaces, like a human head [2, 16, 21, 34].

Another suggestion made by prototypes is the use of a dark surface made of foam or a silicon, both for reducing ambient light noise, and also for preventing sharp structures to make contact with the tissue [21, 34, 48].

When using the NIRS at the patient's head one usual approach is by using a hat or a knit to keep the optode probe fixed throughout the experiment [34, 48].

2.4.6 Amplifiers

The amplification of the signals is an important step before converting the analog signals to digital ones in order to reduce processing errors.
**Transimpedance Amplifier**

The Transimpedance Amplifier (TIA) is very common to be used in the designs of CW systems. It transforms and amplifies current signals coming from PDs into voltage. This is good because the circuitry can be reduced when we are working with voltage instead of current [21]. As said on the detectors section there is an available PD with an integrated amplifier that is used by several prototypes (OPT101). It saves even more space for not needing extra circuitry for the TIA [34].

**Operational Transconductance Amplifier**

The Operational Transconductance Amplifier (OTA) is a preamplifier for low-amplitude signals before filtering or remodulating the signals in order to not remove these elements from the final NIRS signal. It is useful when the analysis is applied to an application that has very subtle chromophore concentration changes like sleep analysis [32].

**2.4.7 Power Supply**

All wireless NIRS systems require a battery as the power source, some of the systems like Kim et al's prototype used an additional battery for supplying high-pulsed currents for the LEDs in the group’s system [2]. Muehlemann et al suggested a power supply circuitry to avoid crosstalk, using shielding planes, passive filters and switched power circuit. This group has used a 3.7 V recharchable Lithium-Polymer (LiPol) battery with safety circuit integrated, the battery lasted 180 minutes for data collection. [16] Safaie et al proposed a drive programmable current in order to increases the calibration accuracy and also used a high-efficiency switch-mode power supply, with that reduces the power consumption and noise due the supply and increases the overall SNR [21].
2.4.8 Processing Units

The processing unit is used for controlling the hardware (ADC, power management board, the unit could also have a data acquisition board) and processing. The processing units usually consist of microcontrollers [2, 16, 22, 28, 49], such as an ARM architecture, or FPGAs [21, 50]. The microprocessors have a pre-defined function pins and microprocessors are cheaper than the FPGA which can have each pin edited to a desired function, however one should program basic stuff as memory, when one is using an FPGA.

2.4.9 Wireless Options

There are some popular wireless options such as: wi-fi, Bluetooth, ZigBee. It is important to know the project’s application before picking the most suitable architecture [51].

Wi-fi advantages are the flexibility, mobility, the possibility to have high data rates and works from medium to long ranges (50-600m) [51]. Bluetooth is said to work with medium data rates and the distance that the device works varies from 10-100m, the Bluetooth’s advantage regards to the device’s low price [21, 51]. Zigbee stands out to its low data rate, reliability, low price and low power consumption, the disadvantage is on the small distance it works 9-100m and a limited amount of data the Zigbee can read and write [51, 52].

One option for wireless medical devices is the Bluetooth 4.0 or Low energy Bluetooth, that is able to transmit small package of data at a short range using less power and a less expensive hardware than older Bluetooth versions [53, 54], it is really useful for saving the battery to work to longer times, and make medical devices more suitable to home care use [53].
2.4.10 Wireless NIRS systems Examples

Commercial systems

Scholkmann et al [27] showed a list of NIRS commercial systems, and those below are the ones that are wireless:

- Artini's Portalite [55]
- MRRA's Genie [58]
- Hitachi's WOT [56]
- NIRx's NIRsport [59]
- fNIRS Devices’ fNIRS1100w [57]
- Spectratech’s OEG-SpO2 [60]

Table 2.4: Commercial NIRS systems characteristics

<table>
<thead>
<tr>
<th>Device</th>
<th>Size(cm)</th>
<th>Weight(g)</th>
<th>Price(US$)*</th>
<th># Sources/ # Detectors/ # Max. Ch.(s)</th>
<th>λ(nm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Portalite</td>
<td>8.4x5.4x2 d. 88</td>
<td>≈11,000</td>
<td>3/1/3</td>
<td>760/850</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5.8x2.8x0.6 p.</td>
<td>-</td>
<td>≈11,000</td>
<td>705/830</td>
<td></td>
</tr>
<tr>
<td>WOT-100</td>
<td>15x11.5x6.2 d. 650</td>
<td>≈32,290</td>
<td>1/1/16</td>
<td>650</td>
<td>705/830</td>
</tr>
<tr>
<td></td>
<td>26x28x9.2 h. 650</td>
<td>(LD)</td>
<td>-</td>
<td>730/850</td>
<td></td>
</tr>
<tr>
<td>fNIRS1100w</td>
<td>8.5x2.5x10.5 200</td>
<td>≈3000-</td>
<td>1/2/2</td>
<td>730/850</td>
<td></td>
</tr>
<tr>
<td>Genie</td>
<td>22x11x9   -</td>
<td>-</td>
<td>1-16/1-32</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>NIRsport</td>
<td>10.5x17x4 350</td>
<td>-</td>
<td>8/8/1</td>
<td>760/850</td>
<td></td>
</tr>
<tr>
<td>OEG-SpO2</td>
<td>11.4x3x2.5 d. 250</td>
<td>-</td>
<td>6/6/16</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

* Prices estimated by a commercial site[61]
** d. processing unit
*** p. application probe
**** h. headset

Table 2.4 summarizes the commercial NIRS systems main characteristics. Portalite stands out for having a shorter size and weight than the proposed prototype in this system, additionally Portalite has more channels than the prototype in this paper and an 8-hour-lasting battery on it. [55], differently from the others that lasts from 1 - 2.5h and a Bluetooth is able to send data to a device 100m away differently to the other devices that sends
data through wi-fi connections [56, 60]. However Portalite’s price is much higher than the proposed device in this paper, US$ 11,000 [61].

fNIRS1100w shows potential to be a great challenger to the device designed in this thesis, fNIRS1100w’s size is shorter but the fNIRS1100w is weight is slightly higher than the proposed device [57], all the others size and weight are no match to the proposed devices. The number of available channels is a factor that influence the price, since the cheaper device is the one with 2 channels [57], while the one with the most channels is the most expensive, however WOT is the only one to use LDs instead of LEDs like all the others. WOT-100 is the only NIRS commercial device that provides free analysis software, differently from the others devices that did not cite if their softwares add cost to the device's price [56]. The wavelength choosing was done with the help of the last column of the Table 2.4.

Literature Prototype systems

A list of literature groups that designed similar NIRS devices as proposed to the designed system of this thesis, as well an overview of NIRS devices’ characteristics is showed on table 2.5

- Adhika et al [28]
- Atsumori et al [62]
- Achigi et al, [32]
- Flexman et al [49]
- Guo et al [63]
- Kim et al [2]
- Muehlemann et al [16]
- Piper et al [64]
- Safaie et al, 2013 [21]
- Sawan et al [50]
- Yurtsever et al [5]
- Zhang et al [22]
Zhang et al's and Yurtsever’s prototypes also used the embedded system (WinCE, Embedded Visual C++), and a microprocessor for developing a NIRS system [5, 22]. Adhika et al uses a Beagleboard that also uses an ARM processor but the device uses the Linux that comes deployed on the beagle board, the other groups’ designs are similar however they have not added an wireless feature in the group’s device, and the other groups have not designed their own Linux [28], just some modules, those literature work uses similar procedure that was used in the prototype described in this thesis. The other characteristics are discussed in the previous sections.
Chapter 3 - Embedded Linux

3.1 Embedded Linux Story

The first known embedded system was the Apollo's guidance computer at early 1960s, EL had the first application later, in fact, the Linux OS was created after that, specifically in 1991 by Linus Tosvalds, a finnish student at the time. This first Linux was written in Intel iA32 architecture to a Motorolla processor, Linus found too difficult the process of porting the system in the processor, so he re-thought the Linux to be easier to be ported to several architectures, and this idea together with several contributors worldwide due to GNU General Public License (GPL) made Linux more and more popular and robust to run in several systems and with several peripherals'support, and also several Linux distributions (Distros) [8, 13]. The Linux story can be seen in a video with less than 4 minutes celebrating 20 years of Linux: Embedded Linux Story [65].

The first registry of the use of the term EL is on a research project made by Michael Barabanov and Yokaiden, in 1996. In the following years 1997-2004, the use of Linux in embedded systems has grown significantly, with the help of important companies such as Samsung, Atmel, Intel, ARM, IBM, HP, among others. The procedure of building EL became more standardized, what also helped the popularity of this concept grow [8]. By the year of 2005 LinuxDevices.com registered about 300 commercially products using EL, ranging from cell phones, and TV sets to outer space applications [7]. Tools to facilitate the creation of custom EL some applications were created on the last years such as Buildroot,
that has been started to use in 2005 in order to create root filesystems that runs Linux to several architectures [67]. There is also Yocto Project that was created in 2010 with the purpose of facilitating the procedure of creating a custom EL for the desired application of the user [68].

3.2 Embedded Systems

3.2.1 Alternatives to Embedded Linux

FPGA

FPGA is a reconfigurable field-programmable digital hardware that can assemble the system’s Inputs, Outputs (I/Os) and logic blocks (cells) as the developer wants, providing countless possibilities to the developers. FPGA’s advantages towards the microprocessors are that the FPGA can operate at a high operation speed and the FPGA has a large number of components to be reconfigurable, FPGA is indicated when there are several processes to run in parallel. However there is one considerable difference between the FPGAs and microprocessors that is the price, and despite the FPGAs process have been reducing the microprocessors are still cheaper [9, 69, 70].

One Linux Distribution widely used in FPGAs is the μClinux (“you-see-linux”) because it does not need a memory unit to run, what is the case of many FPGA cores, and μClinux also provides C libraries, command shells and Unix system calls, without taking out the reconfigurable property of the FPGAs, some examples that are using μClinux follow [69, 71].

FPGA was used by Safaie et al [21] to build a wireless NIRS system, but the group does not enter in much details about the programming the group used, they just commented what the software running on the FPGA did.
Windows CE

Windows (WinCE) is a Windows OS for minimal computer and embedded systems. WinCE uses the same idea of having a kernel like EL. WinCE is usually intended for systems with low available memory, and 32-bit processors (It does not offer support to several architectures like Linux OS does) and WinCE can run under 1 MB of memory. Windows CE development is done with a configuration tool called Platform Builder [22, 9, 72]. There are some other embedded Windows available as Windows Mobile and Windows XP embedded, that are more complex than the WinCE [9].

WinCE is used by Zhang et al to build a wireless NIRS system [22], the group has used the Platform Builder 4.2 to configure, compile and import it into a Microsoft Embedded Visual C++ 4.0, And This group also designed their user software responsible for the serial communication, the display and the data processing with the Embedded Visual C++ [22].

3.3 Anatomy of Linux

A Linux system is usually divided in 3 main parts: The Bootloader, the Kernel and the Root Filesystem. This typical system is illustrated in fig. 3.1 [12]. These parts can also be called as images. Further information about each component is going to be given in the following subsections.

![Figure 3.1: Typical Linux architecture components. The bootloader, the boot parameters, the kernel and the root filesystem](image_url)
3.3.1 Bootloader

Despite of running for just a few seconds the bootloader is a very important component to an EL System Startup, the bootloader’s main functions are: programming the system memory controllers, initialize processor caches, enable hardware devices, implement support for network booting infrastructure, and loading the kernel image and the filesystem into the system. The bootloaders perform the activities extensive firmware activities in Windows systems do like the BIOS, UEFI and OpenFirmware [12].

The most famous bootloader for EL is the U-Boot, the “Universal Bootloader”. U-Boot is an open source bootloader created and maintained by the Wolfgang Denk of DENX Software Engineering (Germany, Munich). U-Boot is considered the most flexible and the most actively developed and documented bootloader, this bootloader has several configuration capabilities and it is available for free. U-Boot is really flexible allowing people to work with several standalone development boards with architectures: PowerPC, Arm, x86, Motorola, MIPS, NIOS among others. It also allows one to set where a boot to USB, File Transfer Protocol (FTP), Network File System (NFS), and other boot capabilities. It also can mount several kinds of filesystem like: JFFS2 (Linux); Cramfs (Linux); FAT (Windows); ext2, ext3, ext4 (Linux) [12].

3.3.2 Kernel

The Kernel is the most essential component of a Linux system, the Kernel is responsible for managing how the hardware use of the system, scheduling the system’s tasks use and managing the correct resources to make the hardware work. The Kernel can communicate the system to the hardware by not necessarily using a direct communication [12].

One important feature of the kernel is that during the development the developer can remove support to certain hardware and filesystems that are not going to be used, or add this support that was not available before, and one can add the support to a hardware modulable,
in other words the modulable hardware is the capability to load a hardware driver only when hardware is plugged on the Linux Host. A desktop Linux (Ubuntu, Debian, Fedora, for example) can support a wide range of hardware but as a new Kernel can be edited the Kernel can be optimized only with the needed capabilities of a system, and that’s the characteristic of an EL. By using a more optimized kernel a better performance of the system can be achieved and an easier and more precise debug of the system is also achieved in order to save time and possible errors to happen during development [12].

3.3.3 Root Filesystem

In the end of the Kernel loading the Root Filesystem is mounted. The root Filesystem is basically the user space files containing, all the binaries, libraries, device files, Kernel modules, system applications and the custom applications on the system to make it run. The root top-level file directories for EL are very similar to those in a desktop Linux [12].

3.4 Development considerations

3.4.1 Components of a design environment and their Interaction

During development two actors can be stood out the Host and the Target. The Host is where the development is going to take place, in other words, the Host is the computer station which the developer will use, when building an EL the Host should be a Linux workstation for avoiding compatibility issues and for using the several tools that are already available for developing EL systems (toolchains, Buildroot, cross-compilers etc.), even though there are tools that help Unix and Windows stations to perform the development, Linux is the most reliable pick for this purpose. Using Linux as the Host is also a good thing for beginners for getting familiar with Linux and how to solve issues [12].

Target is simply where the EL is deployed or is going to be deployed. There are
some combinations on how the relation between the target and host can be made it depends mostly on the target hardware one has, but one’s design can use more than one configuration and these configurations can be changed over time [12], these setups are:

- **Linked Setup** - In this configuration the *Host* is linked to the *Target* through a physical connection (USB, RS-232, Ethernet) or a remote connection (FTP, NFS). This configuration is the most common one, and this Linked Setup is used for debugging purposes, the main property of this configuration is the sharing files from *Host* to target and vice-versa, the Linked Setup is also possible to mount the *Target* system in the *Host* and manipulate the *Target* files from the *Host*. The scheme of the Linked Setup is illustrated by fig. 3.2(a) [12].

- **Removable Storage Setup** - In this setup the development performed in the *Host* is transferred to a storage media, (SD Card, CD-ROM, DVDs, Disk Floppies) usually the *Target* system has a very simple bootloader, and the storage media has another bootloader to boot everything else on the *Target* system. The scheme of the Removable Storage Setup is illustrated by fig. 3.2(b) [12].

- **Standalone Setup** - In this one the *Target* is much likely a PC working station because all the tools are already embedded on the *Target*, the needed tasks on this setup are defining the project’s needs by the developer. In some cases an actual Linux distro (Like Fedora, Debian) can be installed in this standalone target board and then the developer remove the unnecessary tools and add what he intends for his/her application. The scheme of the Standalone Setup is illustrated by fig. 3.2(c) [12].
Figure 3.2: EL development setups. (a) Linked Setup, where the Host and Target are connected in some way during development (b) Removable Storage Setup, it is the development that happens on the Host and by a storage media it is transferred to the Target the (c) Standalone Setup, where the development happens directly at the Target. Adapted from Yaghmour et al [12]

3.4.2 General Public License

The General Public License (GPL) is the license which Linux was built, this leads to a popular belief that Linux is free, although the GPL states that when they speak about free software they mean software with free access and not software with no cost. There is an example that says that Linux is not free as “free beer”, Linux is free as in “free speech”.

Having a software that's free as Linux, people can have the following important outcomes: Linux grants the possibility for the user to study, use, modify, and share his/her modifications on the original code. And this procedure definitely has a cost because of the development, deployment, maintenance and support efforts for this software. This license also grants the right of not sharing one’s code, but in this case the contributor cannot sell his/her contribution before he/she shares it [73, 7].

3.4.3 Design Methodology

Yaghmour et al [12] suggests a design methodology in order to build someone’s own EL for a target. There are 4 main steps as follows:

1. **Determine system components** - This phase consists in determining all the hardware and software that is going to be used in this design in a list, like planning a shopping
list [12].

2. **Configure and build the kernel** - This phase consists in determining which Linux kernel version is going to be used and to look very carefully and study each option on kernel editing having in mind the system requirements [12].

3. **Build the root filesystem** - This phase can be run in parallel with the kernel building and configuration, this phase consists in having a system with all the libraries in binaries necessaries to run one’s system, the building procedure can either be done by having a system with all components one wanted or by having a system and adding/removing tools and libraries as one develop it (using an iterative method). Taking notes is important on the iterative method so one can build a “clean” and optimized root filesystem by default [12].

4. **Set up boot software and configuration** - During this step the bootloader, the kernel and the root filesystem are brought together to the *Target*, this step mainly took place at the bootloader where one is going to set the new kernel and root filesystem on the bootloader’s setup, and where the images are going to boot from, NFS, Flash memory, SD card, or somewhere else [12].

### 3.4.4 Relevant Concepts

**Cross-Compilation**

When one is using a host and one needs to compile a program or an OS to a target system with different architecture than one’s host, the compiling process needs tools (called toolchain) designed for the *Target’s* architecture. For example let's suppose one needs to compile a software to an ARM target using a x86 architecture host, in this case one needs to cross-compile x86 to ARM by using an ARM toolchain.[9, 12]
Toolchain

Toolchain is the set of tools needed for Cross-Development like assemblers, linkers, compilers (such as GCC) and other building tools (such as GNU build tools) [8, 9]

Linux Distribution

Everyday people refer “Linux Kernel” as the “Linux Distributions” we know like the famous ones: Debian, Red Hat, Ubuntu, CentOS and Fedora. In fact, Linux Distribution consists on a prepackaged set of files and softwares, with an installing procedure to provide an user a Kernel/Filesystem for a generic purpose[12].

Source version control and git

Source version control is a common technique to track changes on code, mostly when there is a development team on a project. There are several tools in order to do the version control automatically, one of the most popular ones to do this is the git, used in this project, all work done with git can be seen on the Appendix. Git can be used to clone, download and upload change notes or code on its repository, called github. Github is handful during development in order to share one’s work, to have a backup, and to have a controlled multi-site development and to track changes in code and who made that change [7, 12].

3.5 Building tools

3.5.1 Popular embedded development boards

The Linux.com and LinuxGizmos.com had a 10-day survey in 2014 asking which Linux/Android development board were the Linux developers’ favorite and the result was that RPi and BBB were the ones respectively. Coming in the next positions on the top 10
were the Cubicle Boards (2 and 3), and the Odroid-U3 board. The Cubicle Boards and the Odroid-U3 also have big independent development communities like the winners, and they have a bigger support to Android. The RPi’s clone: Banana Pi appears in fifth on the list as well some boards that are getting more popular as Intel's Galileo and the udoob board [66].

Figure 3.3: Popular development boards considered for the project, where (a) is the BBB and (b) is the RPi

RPi and the BBB that were considered to this project (fig. 3.3), both were considered because these boards are relatively easy to be acquired. Some relevant comparing factors for the development of this project are showed on the table 3.1. The number of General-purpose Input/Output (GPIO) pins are higher on the BBB as well the processing power, the RPi is smaller and cheaper, however BBB's lighter, the community support to RPi is bigger and older than the BBB [74, 75].

<table>
<thead>
<tr>
<th>Feature</th>
<th>BBB</th>
<th>RPi</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPIO</td>
<td>69</td>
<td>40</td>
</tr>
<tr>
<td>Processor(MHz)</td>
<td>1000</td>
<td>700</td>
</tr>
<tr>
<td>Dimension(mm)</td>
<td>(89x55x15)</td>
<td>(56x85x1)</td>
</tr>
<tr>
<td>Price ($)</td>
<td>55</td>
<td>40</td>
</tr>
<tr>
<td>Weight(g)</td>
<td>40</td>
<td>42</td>
</tr>
<tr>
<td>Year of release</td>
<td>2013</td>
<td>2012</td>
</tr>
</tbody>
</table>
The BBB’s origin differs from the RPi, while the BBB is the american answer to the success of the british RPi [74]. The purpose of each one also differs the RPi is mainly intended on education in order to make young people get interested on how computers work and how to program, while the BBB is intended to EL developers with a higher hardware connection capability and a more powerful processor [76].

BBB is also intended to projects of low complexity in circuitry but high computation complexity, BBB is also relatively easy to design a cape for [10]. A cape is a circuitry that perfectly fits on the BBB’s GPIO pins. Mostly because of BBB’s GPIO advantage and the processing power the BBB was chosen for this design.

3.5.2 BBB overview

In this section the most relevant features of BBB regarding this work is going to be showed [10, 77].

Features on the board

On fig. 3.4 the main components used during the project are going to be described on the following paragraphs. The BBB usually comes with an Linux Angstrom distro deployed in the board.

Firstly talking about powering the board, the board can be either powered by the 5VDC input, or by the battery connections or by the USB client (that is a miniUSB port). The default BBB, comes with a miniUSB-USB cable that can be used to both turn on and access the board via a host computer. Once the procedure to add a battery is done, the miniUSB port is disabled to both access and power the board. The 5VDC connector only powers the board. The maximum current that can be used to power the board is 2A, when using a battery or the 5VDC to power the board. When the board is powered, the BBB’s power LED (Between the the 5VDC and the Ethernet Port) is activated [10, 77].
Secondly accessing the board via host is essential to debug, use, and test the projects at the board. In order to access the board that, there are 3 options, the first one is by using the miniUSB port and then accessing BBB through Secure Shell (SSH) or via serial port, the other option is accessing the BBB through the local network by using an Ethernet cable connected to the BBB, lastly there is the Debug Serial Header for accessing the board. The Debug Serial Header can be used even when the USB client is disabled. An important detail to know is that the J1 pin on the Debug Serial Header is the ground (GND) pin as showed on fig. 3.5 (a), the cable to be connected on that is a FTDI cable showed on fig. 3.5 (b). One detail to remember is the BBB is not connected automatically the board is usually connected via a terminal software like PuTTY, Minicom or Picocom [10, 77].

Thirdly there are 3 switches on the board: the Boot Switch that is used to either booting from the MicroSD card by pressing the button together with the Power Switch, or if the MicroSD card slot is empty BBB will try to boot from the USB client port and then from the USB port. The Power Switch is the button that powers the board, if one holds this button when the device is on, the button forces to shutdown BBB, if one presses the button without holding the button, when BBB is on, the shutting down command is called. The
Figure 3.5: Debug Serial Header connection position and cable. (a) Right position to connect the FTDI cable on J1’s Debug Serial Header GND (connect it with the black wire of the FTDI cable). (b) The FTDI cable. Adapted from Molloy [10]

reset switch either restarts the system when the device is on or force shutdowns the system when the system crashes [10, 77].

Regarding a BBB’s other components, the Ethernet is used to download during development if a package is needed to be installed, and to connect the board to the network the USB Host is where both the Bluetooth Dongle or pendrives are connected to the BBB to gather the logfiles. The user LEDs indicates buffer activities and when the system is booting up or shutting down, when the user LEDs are all off right after turning the BBB on means that the system stopped on the U-boot because the system was not shutted down properly. Other essential components for this design are the P8/P9 Expansion Headers that are the points where, GND, power, digital I/O and analog input pins are, besides the structure to fix the capes designed to this board [10, 77].

Issues to avoid on the board

BBB has some limitations that must be aware by the developers in order to keep the board working functionally right. Here is some list to be aware of [10]:

- BBB should always be shutdown by using the shutdown button or shutdown commands like “halt” in order to avoid damaging components or files on the system.
• BBB Should use some “feet” on the board to avoid direct contact with a surface and possibly damage or short pins on the board.

• One should not place the board on a metal surface, it will short the board’s pins.

• Before turning on BBB one should check if there is a short between P8 and P9 Expansion Headers.

• 8mA currents can damage GPIO pins, One should try to work within a max range of 6mA in order to avoid burnt pins

• Digital pins can read a maximum 3.3V, while the analog ones 1.8V.

• One should not apply voltage or current to P8/P9 header pins if the board is off.

3.5.3 Buildroot

Buildroot [67] is a framework for EL building, buildroot came with the idea to simplify and automate this complex process, because rebuilding the Linux is needed for several times during the Design [78]. On fig. 3.6 buildroot interface is showed.

Figure 3.6: Buildroot Interface Example. Adapted from Petazzoni [78]
The process of building an EL is by building a toolchain and performing cross-compilations from the Host to the Target. Building a Linux is possible to be done by hand but this activity is time consuming and in much of the times is frustrating because the amount of errors involved in a very complex task, and errors are constantly corrected in Buildroot on every new stable release. Buildroot automates all this process by having makefiles, patches, packages, target libraries for building the environment and the resulting Kernel, Bootloader and Root filesystem images of the EL. Buildroot also allows to create one’s own patches and then use these custom patches with the Buildroot to build EL projects [9, 73, 7].

3.5.4 YoctoProject

Yocto Project [68] is an open source infrastructure that provides tools, templates, toolchains and software that help developers to create their own custom EL. Yocto counts with companies, communities and people working indirectly together, providing a lot of material to give support to developers in activities as reusing previously built utilities, software, libraries among other components. Yocto Project prepares the build environment for the user reducing time and efforts of the developer in this phase of the project. Yocto Project was created in 2010 and is supported by the Linux Foundation Community [68, 79, 80].

Some examples of built-in tools on current versions of the Yocto Project are:

- **Poky** is an integrator of BitBake, OpenEmbedded-Core and metafiles available to build the custom EL. Practically, Poky is the file required to be downloaded at the yoctoproject.org to start the development. Poky's main purpose is to provide all the features that an EL developer will need [79, 80].

- **BitBake** is responsible for managing configuration files, dependencies, the packages’ recipes and organizing the tasks in a way that there is going to be no dependency errors and that the code generation and compilation is going to take the least time possible. BitBake also tries to reduce the build time by maximizing the use of pro-
cessing resources, like choosing how many cores of the Host's processor would be allocated for this activity [79, 80].

- **OpenEmbedded-Core** is a metadata collection to provide support for five popular microprocessor architectures in embedded systems ARM, x86, x86-64, PowerPC, MIPS and MIPS64 [79, 80], in this project ARM is the architecture that's going to be used.

- **MetaFiles** are the files containing Python and Shell Scripts that will provide the instructions on how to build packages depending on the configurations set on the BitBake. There is a possibility that one’s own files are used to create one’s own or reuse other users’ packages for the design [79, 80].

- **BSP** The Board Support Package(BSP) is a collection of files that supports a hardware platform, a hardware device or a series of hardware devices. BSP also can contain a series of softwares that can be essential or optional to a specific design [81].

- **Hob** is an user-friendly interface to BitBake, helping the developers to customize images, gather help explaining the packages, and after the images are done the developer can simulate the built EL on the QEMU [79, 80].

- **QEMU** is a free and open-source machine simulator that is able to provide hardware visualization without involving the actual hardware. QEMU supports the same hardware architectures supported by the OpenEmbedded-Core [79, 80].

YoctoProject tool was chosen to design the EL in this project due an available support to BBB.
3.6 Embedded Linux in Biomedical devices

Some studies have already used EL in order to develop Biomedical devices, such as Lee and Kim [82] that created their own EL responsible to acquire, store, process and analyze medical imaging, those images could be displayed real-time on a computer station and the acquisition device could be portable to several scanning devices.

Da Silva et al [83] designed a portable heart monitor with a BBB. This Group’s device is able to apply a series of algorithms to identify heart diseases, data could be transmitted online by this group’s device.

Paez et al [84] developed a system to evaluate the performance of archers with a series of sensors, he also uses a BBB with a EL on the created system.
Chapter 4 - Methods

4.1 Resources used

In this section the materials used, the items’ prices, how many of each item was used, and where the items were acquired, are informed on tables 4.1 and 4.2.

Table 4.1: Items list to develop the system, the quantity, prices and where these items were acquired from

<table>
<thead>
<tr>
<th>Item</th>
<th>Unit</th>
<th>Price(US$)</th>
<th>Available from</th>
</tr>
</thead>
<tbody>
<tr>
<td>BBB</td>
<td>1</td>
<td>55</td>
<td>Adafruit</td>
</tr>
<tr>
<td>LiPol 3.7V 2000mAh battery</td>
<td>1</td>
<td>12.50</td>
<td>Adafruit</td>
</tr>
<tr>
<td>Bluetooth USB module</td>
<td>1</td>
<td>10.95</td>
<td>Adafruit</td>
</tr>
<tr>
<td>IR 850nm emitter</td>
<td>2</td>
<td>2.32</td>
<td>Digikey</td>
</tr>
<tr>
<td>IR 765nm emitter</td>
<td>2</td>
<td>5.96</td>
<td>Digikey</td>
</tr>
<tr>
<td>OPT101 photodiode</td>
<td>2</td>
<td>14.18</td>
<td>Digikey</td>
</tr>
<tr>
<td>Diode</td>
<td>2</td>
<td>-</td>
<td>University’s Lab</td>
</tr>
<tr>
<td>Resistor</td>
<td>6</td>
<td>-</td>
<td>University’s Lab</td>
</tr>
<tr>
<td>Capacitors</td>
<td>2</td>
<td>-</td>
<td>University’s Lab</td>
</tr>
<tr>
<td>Colored wires</td>
<td>20</td>
<td>-</td>
<td>University’s Lab</td>
</tr>
<tr>
<td>Breadboard</td>
<td>1</td>
<td>0.5</td>
<td>Digikey</td>
</tr>
<tr>
<td>Header Male Pins</td>
<td>94</td>
<td>-</td>
<td>University's Lab</td>
</tr>
<tr>
<td>Solder</td>
<td>1</td>
<td>-</td>
<td>University's Lab</td>
</tr>
<tr>
<td>1/8”Neopropene Sheet</td>
<td>1</td>
<td>3.79</td>
<td>Zoro</td>
</tr>
</tbody>
</table>

SubTotal: 105.20
Materials on the system

On the table 4.1 the items present on the system are highlighted, items came from the online stores: Adafruit, Zoro and Digikey. Adafruit is a site that mostly sell items for electronics hobbyst, Digikey is a very popular electronics online store and Zoro is an online industrial supplies store. All of the items that were bought can have lower prices when bought by negotiating greater quantities, that is not the case in this project where the items were bought in low quantities for designing a single prototype. Regarding these items that were not priced, these items were available at the University’s Lab for use, and they are items that are easily found in simple electronic labs, and even tough if those items were added on the project they would not increase the project's budget. Every single item on table 4.1 were used for the design of a single prototype.

Materials used during development

The table 4.2 refers to the items used for developing purpose only like the FTDI cable (used for Debugging), or items that could be used in building replicates of this prototype if a large scale of devices were produced such as the battery charger, the microSD card (used to deploy the EL Image), the soldering iron, the wire cutter, screwdriver, sandpaper, scissor and pliers.

Table 4.2: Development Items and tools used in this project, the needed quantity, price, and where this items were available from

<table>
<thead>
<tr>
<th>Item</th>
<th>Unit</th>
<th>Price(US$)</th>
<th>Available from</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 GB microSD</td>
<td>1</td>
<td>7.95</td>
<td>Adafruit</td>
</tr>
<tr>
<td>USB LiPoly charger</td>
<td>1</td>
<td>12.50</td>
<td>Adafruit</td>
</tr>
<tr>
<td>FTDI serial cable</td>
<td>1</td>
<td>17.95</td>
<td>Adafruit</td>
</tr>
<tr>
<td>Soldering Iron</td>
<td>1</td>
<td>-</td>
<td>University's Lab</td>
</tr>
<tr>
<td>Tools (wire cutter, screwdriver, sandpaper, scissor, pliers)</td>
<td>-</td>
<td>-</td>
<td>University's Lab</td>
</tr>
<tr>
<td><strong>SubTotal</strong></td>
<td></td>
<td><strong>38.40</strong></td>
<td></td>
</tr>
</tbody>
</table>
4.1.1 Choice of Materials

The LEDs (765 and 850nm) were the picked ones because the their small size, low power consumption and due not having the need of a complicated circuit. The wavelengths were chosen by taking a look at table 2.4 and table 2.5 as a matter of comparison and the concept discussed to reduce the crosstalk between the measurements on both wavelengths discussed by [45, 46]. An overview of these LEDs are showed on table 4.3.

Table 4.3: Specifications of the LEDs used, from [85, 86]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>765nm LED</th>
<th>850nm LED</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>M TE2077N1-R [85]</td>
<td>TSGH6400 [86]</td>
</tr>
<tr>
<td>Diameter (mm)</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Spectral Bandwidth (nm)</td>
<td>35</td>
<td>40</td>
</tr>
<tr>
<td>Forward voltage (V)</td>
<td>1.55-1.9</td>
<td>1.5-1.8</td>
</tr>
<tr>
<td>Max current (mA)</td>
<td>50</td>
<td>200</td>
</tr>
</tbody>
</table>

The OPT101 showed good potential to both size and price, and the additional TIA on the detector was also positive to save space on the circuit. Regarding to OPT101’s technical specifications, the detector has a responsivity of 0.45W/A, the OPT101 works supplied by a positive voltage ranging from 2.7 - 36 V, the PD’s responsivity is higher to NIR light (700-950nm), and this detector can work up to a frequency of 14kHz and the PD area is 0.09 x 0.09 in (2.29 x 2.29 mm) [87].

4.1.2 Software needed

As said on previous sections, the first software necessary in a host system when designing an embedded system is the terminal emulator. During this project the two mostly used terminal emulators were putty and picocom, that are used in Windows and Linux host systems respectively.
PuTTY

PuTTY is a popular terminal emulator that can be used in Windows systems to connect a computer to other system through raw, telnet, ssh, Rlogin and serial. In the case of accessing the BBB, the first necessary step is to install the BBB drivers and the FTDI drivers to access the board on the Host, once the board is connected through the miniUSB-USB cable provided together with the board. The board is powered and the FAT files are accessed, in a folder called Drivers select the file according to the architecture of the computer 32 or 64-bits. Then in order to access the board via PuTTY we should select the SSH option and then write 192.168.7.2 as the IP and 22 as the port as showed in fig. 4.1 then press connect. When the board connects a terminal window is shown and when the login is requested “root” is written and then enter is pressed when the password is requested.

Figure 4.1: PuTTY terminal emulator settings to connect to BBB via SSH

Regarding the FTDI cable, used to access the board, in a Windows system it is necessary to install the driver (available at this link FTDI Driver [88]). Then look with a left click in “my computer” option >>>“Manage” and check at which COMx the FTDI cable
connected is, where “x” is the COM number. Now in PuTTY choose the option “Serial” and then write the COMx with the number x checked before and use the BAUD Rate as 115200, then press connect, and then write “root” when login is required. If one does not want to keep always configuring PuTTY it is possible to save this setup on the button save and then just double-clicking this saved option will access the BBB without needing to set the parameters.

**minicom**

Minicom is relatively simpler to configure in a Linux system, Minicom is also a terminal emulation program analogue to Putty but it is used in Linux host systems, In order to install it in a debian version of Linux just open an terminal (fig. 4.2) use the following command:

```
sudo apt-get install minicom
```

![Figure 4.2: Linux Ubuntu terminal](image)

In order to configure the Host system, minicom is needed to be called in configuration mode as the root, by using the command below, and then on minicom’s options the
port used should be set as /dev/ttyUSB0 (It could differ on different systems), turn off the hardware and software flow control and set the baud rate as 115200. In the end “save this configuration as dfl”.

```bash
sudo minicom -s
```

And then minicom is ready to work, in order to call minicom again, in a terminal window use the command below, where the –term=vt102 is a flag to activate a minicom’s mode where the system has fewer bugs when showing text.

```bash
minicom –term=vt102
```

**picocom**

Picocom is another Linux terminal emulator, it is simpler to use it and more complex to call and configure picocom but it does not have bugs showing text like minicom, in order to install picocom use the command below.

```bash
sudo apt-get install picocom
```

In order to run picocom to access BBB the configuration to access the board is used in the same line to call picocom , where the flag -b is used to set the baud rate 115200 and the flags -r and -l are used to set the port /dev/ttyUSB0. The command is showed below.

```bash
sudo picocom -b 115200 -r -l /dev/ttyUSB0
```

**nano**

The program nano is a light text editor that can be used in a Linux terminal to edit and create files, during the development nano is mostly used to create and adjust python and bash scripts.
The Linux package bluez is used for controlling the Bluetooth devices. In this package there are some tools that work to initiate the Bluetooth, make the Bluetooth work as plug-and-play, create a Bluetooth port to make the device accessible and scan for devices.

Python is both the program that is used to run python scripts/codes and Python is also a programming language used during this project. Python has several support and packages, to work in several opportunities.

Bash is like python, bash is both a program to run bash scripts and a programming language. Bash is more common on init scripts on Linux and to perform low-level tasks on Linux.

Git is a software in Linux to upload one’s code in a git repository, and further make a version control of code files uploaded there.

4.2 Overview of the prototype

4.2.1 Diagram

On fig. 4.3 the system overview is showed summarizing how the prototype works: The NIR LED sources irradiates light into tissue, and the back-scattered NIR light is detected by the PDs that transduce the light into an amplified voltage signal with the help of a TIA,
coupled on the PD. The signal is read at the BBB analog's input, the BBB also controls the NIR sources alternating the power of each source with different wavelength. The processing unit is powered by a LiPol battery, and after processing the signal. The processed signal can be sent via Bluetooth with the help of a Bluetooth dongle to an Android phone which display real time the signal and BBB can record the raw data to be further analyzed.

Figure 4.3: Diagram overview of how the main items interacts to each other. The NIR sources are activated alternately and detected by the amplified PDs both controlled by the processing unit which also process, store and transmit the data through the Bluetooth dongle to an Android phone to display real-time data. The processing unit is powered by a LiPol battery.

4.2.2 Development Plan

The development was organized in 3 main group of activities: the Linux System Development, The Hardware Development and the GUI Development. Those activities not necessarily happened together or in order, the sequence and relation which these activities
had is illustrated on fig. 4.5

![Diagram](image)

Figure 4.4: Development Plan Diagram, containing Linux System Development, Hardware Development and GUI Development phases

### 4.2.3 Chronogram

In order to record the EL system development system needed time, a draft of a Gantt Chart was created by a Gantt Chart Creator software called *GanttProject* [89]. The Gantt chart worked as a guide to organize which step would be the next one during the development.

The first planned chronogram was designed to be 4 month-long chronogram of work to design and document the whole system.
Figure 4.5: Project chronogram on a Gantt Chart designed with GanttProject. On the left table on the Gantt Chart there are the names of the tasks, the beginning and end dates of each task. On the right side of the Gantt Chart there are progress bars that can evaluate how much was done on a task by the use of a timeline

4.3 NIRS system Assembly

4.3.1 Nominal values

Some values were theorized by looking at the components datasheets [85, 86, 87], and to the BBB board itself [10, 77]. The main values that were utilized during the planning part of the circuit are showed on the table 4.4.

<table>
<thead>
<tr>
<th>Voltage supplied to</th>
<th>Value(V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>850nm NIR LED</td>
<td>2.3</td>
</tr>
<tr>
<td>765nm NIR LED</td>
<td>2.3</td>
</tr>
<tr>
<td>Detector's pin1 (PD Power)</td>
<td>3.3</td>
</tr>
<tr>
<td>Analog input pins(Max)</td>
<td>1.8</td>
</tr>
</tbody>
</table>

The voltage applied to the LED is the voltage coming out from the GPIO pins of the BBB, the voltage applied to the detector are the default 3.3 V pins. The analog input pins in BBB can resist to a maximum of 1.8V [10, 77]. In order to create the detector circuit one of the datasheets option was chosen, and besides that a diode was used in order to reduce the voltage level. Regarding to the LEDs, the current needed to be less than 50mA to the
765nm NIR LED [85] and 100mA to the 850nm NIR LED [86]. In order to not harm the LEDs a resistor (the most common ones usually can stand 0.25W power) value that would not overheat or burn the resistor.

4.3.2 Circuits' Design

In order to design the circuits the program LTSpice [90] was used, that is a software used to design circuit diagrams, and to simulate circuits and electronic components.

The first models created to be tested in the LTSpice [90] were the sources circuit, showed in fig. 4.6, since these circuits are simpler and designing the sources circuit with the software was a good way to be familiarized with LTSpice. There were not the exact LED in the program library, so a generic LED and set the component nominal values [85, 86].

![Sources circuit overview designed with LTSpice](image)

Figure 4.6: Sources circuit overview designed with LTSpice

Regarding the detectors circuit the first step was to create the OPT101 detector in the program library, since this component was not available as the other components. When the detector was created the component was used to design the rest of the circuit seen in fig. 4.7.

4.3.3 Circuits' Simulation

The sources circuit were not tested in the LTSpice, they were just modeled in order to be illustrated. They had everything measured in the nominal values section. The simulation
Figure 4.7: Detector circuit overview designed with LTSpice, the red, blue and green marks corresponds to places where the voltage was evaluated during simulation.

consisted on a sinusoidal stimulation going from maximum to the minimal stimulation of the PD. The resulting simulation on the detectors circuit is showed on fig. 4.8, where the red, blue and green curves are indicated in which points of the circuit measurements were made on the fig. 4.7.

The red curve/point represents the output signal from the PD, the blue curve/point represents the output signal after the filter and the green curve/point is the final output that goes to a BBB's analog input, after a reduce in the voltage level.

4.3.4 Testing in a Protoboard

The detector and source circuits were replicated on a protoboard and the amplified voltage has been measured in a well-lit room, in the different spots showed on figure 4.7.
Figure 4.8: Simulation in the detectors circuit running in LTSpice, the red curve represents the output of the PD, the blue curve the signal after the first filter treatment, and the green curve is the output read on the analog pins after the reduction of voltage by the diode. The results are showed on table 4.5.

The top values measured on the protoboard showed on table 4.5 were coherent to the previewed LTSpice's simulation showed on 4.8. And the most important is that the final output (green point) is below 1.8V, that is the desired value to be replied to the design’s final output points to the BBB's analog pins, without damaging those pins.
Table 4.5: Measured voltages on the circuit on a protoboard in a well-lit ambient situation

<table>
<thead>
<tr>
<th>Measurement point</th>
<th>Voltage value (V)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Red point</td>
<td>2.311</td>
</tr>
<tr>
<td>Blue point</td>
<td>2.118</td>
</tr>
<tr>
<td>Green point</td>
<td>1.628</td>
</tr>
</tbody>
</table>

4.3.5 Creating a Cape

First of all the breadboard was measured to fit the board, in order to fit it on the BBB I/O P9/P8 Header pins. Then the breadboard was cut and sanded, after that the source and detector distance were defined as seen on fig. 4.9 and then the circuitry were sketched as seen in fig. 4.10. In order to check if there was enough space for the cape and to the designed circuit a way to have the circuits within the available cape space replied twice, before soldering the electronic components, their “legs” were wired up to fix them in the circuit positions, then all connection points were soldered.

![Image of source-detector distance](image1)

Figure 4.9: Source-detector distance on this design the blue square represents a detector and the red diamond a source

Then all the excess of wires were cut, and every single point in the circuit had a multimeter checking if there were any broken point or a short circuit. The resulting cape is showed on fig. 4.11

In order to have a more organized design, all the GND wires were made black and they were put on the bottom side of the board.
Some extra considerations in this design should be made, when sketching the components positioning the developer needed to keep in mind that the first 3 columns and the last 3 columns were reserved to the I/Os connection pins. The chosen LEDs are taller than the detectors, so the detectors were raised with chip connection pins, these were also practical when the replacement or removal of the detectors were necessary.

4.3.6 Setting the Battery

The process of making the BBB able to work with a battery is summarized by fig. 4.12 and a more detailed procedure is explained by Shabaz [91].

The first step showed on fig. 4.12(a) is to get to know the battery pins on the BBB, the
Figure 4.11: Designed BBB cape (top view)

Table 4.6 shows the function of each pin.

The second step (fig. 4.12(b)) is to solder the components on the pins. On the bottom side of the BBB, at the pins tp8 and tp6 a 0-ohm link (or a jumper) is soldered, on the pins tp5 and tp7 a 10kΩ resistor is soldered in order to simulate a thermistor. On the top side of the BBB male pins for the battery are soldered between tp6 and tp8.

The third step is isolating the male pins from the rest of the circuit, by using an insulating tape 4.12(c). Lastly the female battery pins are connected to supply power to the BBB 4.12(d) to test if the system worked with a battery.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>tp5</td>
<td>Battery connection point(EN)</td>
</tr>
<tr>
<td>tp6</td>
<td>Voltage sense(V+)</td>
</tr>
<tr>
<td>tp7</td>
<td>Temperature sense(TS)</td>
</tr>
<tr>
<td>tp8</td>
<td>Ground(GND)</td>
</tr>
</tbody>
</table>
4.3.7 Recharging the LiPol Battery

The battery is recharged/charged by a battery charger bought at adafruit.com. The procedure to use the charger is connecting the charger’s miniUSB to an USB port via a microUSB - USB Cable, and the battery is inputted to the BATT socket and a yellow LED works by. The charger is connected to a PC via a microUSB - USB cable, and the battery is inputted to the BATT socket by the battery JST cable and a yellow LED to the left of the BATT Socket is lit the following setup showed at fig. 4.13. It takes about 1.5h to charge the battery from 3.97V to the maximum voltage of 4.24V, when the battery reaches the maximum load the charger indicates a green light on “done” written part of the charging board. The default current to the charger is 50mA, the charging current and charging speed can be increased by adding a load at the load socket. If a higher speed to charge a battery
is wanted.

One important info when charging LiPol batteries is that when the batteries overcharges they can blow or catch fire, and one should be aware to proper handle (handle with care) this kind of battery in order to avoid accidents even tough that there is a protection circuit on the LiPol battery being used in this project, if this circuit is damaged the battery can catch fire when it is charging a charge higher than the battery can stand.

![Image of battery setup](image-url)

Figure 4.13: Recharging battery setup, the JST battery pin is connected to the charger on the charger’s BATT pin, and the microUSB of the charger is connected to an USB port to power the charger

### 4.3.8 Rubber Case Design

A rubber case was used to be the dark protection from ambient light, the case can provide a soft application site to the patient and an isolator of the electronics of the device to the patient.
The case design was done by using the Neopropene sheet. Firstly the case’s dimensions were estimated and then the sheet was cut into smaller pieces according to the dimensions of the board and the cape. The top side was designed respecting the BBB’s switches the sources and detectors, a bottom and side sheets showed on fig. 4.14 were used to suit the cape. The sheets were cut thinking on, the size of the prototype, the connecting points to the other sheets, providing enough space where there are buttons and important items on the board such as the detectors, sources and the battery connection pins are not impaired by the case.

Figure 4.14: Cut neopropene sheets used on the case design, the one on the top shows the top sheet of the case, and the 3 below are the left, bottom and the right sides of the case respectively

After mounting the sheets on the prototype the rubber case is in the way showed on fig. 4.15.
4.4 Developing the Embedded Linux

4.4.1 Configuring the Host computer

Downloading dependencies

Firstly in a Linux system, the dependencies to work with the Yocto Project should be downloaded, these dependencies varies depending on the Linux distro used, in This project an Ubuntu 14 system as the host, so the yocto’s mega manual [92] indicated the following packages to be installed in a terminal command below:

```
sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib build-essential chrpath socat libstdc++1.2-dev xterm
```

The other dependency files necessary to other Linux distros are indicated in the Yocto’s mega manual [92].
Setting Yocto’s BSP build environment to BBB

Yocto project provides a BSP to BBB in order to facilitate the development [93]. One can either download the BSP directly at this link, or one can use the following command on the Linux terminal:

```
git://git.yoctoproject.org/meta-yocto -b daisy
```

After downloading this BSP, the file needs to be “untared” and one can either “untar” it by browsing this file and right clicking the file, and then select the “untar” option one wants, or one can use the following command at the Linux terminal where the file is:

```
tar xvzf filename.tar.gz
```

A folder called poky will be created, on a Linux terminal enter this folder and use the following command to set the build environment to use yocto project's tools [94].

```
source oe-init-build-env
```

Depending on the Linux Host the command above cannot work, another also possible command to be used follows:

```
. ./oe-init-build-env
```

Then the system is ready to start the EL building. Before building the Linux system Yocto’s mega manual [92] and Salvador et al [79] suggest the developer to go to the configuration file located at poky/build/conf/local.conf and use any text editor to add the following statement.

```
INHERIT += “rm_work”
```

By adding this statement after a package is built the building files are going to be removed, and consequently disk space is going to be saved, as well bugs caused due no available disk space at the Host are prevented [79, 92].

Other important step is to install a virtual computing network software to test the built images, one remote desktop viewer very popular for Linux is Vinagre, in order to install Vinagre use the command below.
4.4.2 Building Minimal System

A minimal system was built to test if the EL design is going to work or further adjustments in the kernel are needed. In order to build the minimal system, the graphical interface hob was called by using the following command on the Linux terminal after the build environment was ready.

```
sudo apt-get install vinagre
```

![Hob machine and image selection screen](image)

Figure 4.16: Hob machine and image selection screen

The first hob screen as is showed on fig. 4.16. In order to make a system that is testable, the chosen machine is qemuarm, and the image for minimal system is chosen as the core-image-minimal. For building the minimal system to the beaglebone the process is repeated but the machine chosen is the beaglebone and the image-recipe to be chosen is
core-image-minimal, in order to build the system click on the build image.

The package building phase is showed on the fig. 4.17, this phase screen shows the process and which steps are being processed, what was done and which packages are going to be build, the screen also shows errors during the package build. This process can take some hours depending on the Host, but once most of packages are built this procedure is much faster on the next times an EL is built.

If on the screen showed on fig. 4.16 one presses “edit image recipe”, the screen showed would be the one on fig. 4.18, this step is important when one is trying to build an image with the desired or custom packages one wants to add to one’s design. In order to do this custom built, one needs to go to “all packages tab” and search for the packages that one intends to add to the image, some of dependent packages are going to be selected automatically once one desired package is selected. Another possibility the developer has is to remove those packages that will not be needed for the built. Once the packages one
wants to be on one’s project are selected, the button “build image” is pressed to go on a screen like fig. 4.17 where the packages will be built. In case that one does not go through this phase where one edits these packages a custom image is not built or choosing which libs or programs on wants to be on the image.

![Figure 4.18: Package choosing phase on hob](image)

After the packages are built for the minimal design, the next step is to get the images built. When the procedure is finished the screen showed on fig. 4.19 appears on the Host and the images are saved on the folder where the script was run “build/tmp/deploy/images/beaglebone”, these files are showed below and the date and time that the files got done are added to the filenames.

- **Kernel** - uImage.bin
- **Filesystem** - core-image-minimal-beaglebone.ext3
- **Tar Filesystem** - core-image-minimal-beaglebone.tar.bz2
- **Bootloader** - u-boot-beaglebone.img

![Image of Hob](image.png)

Figure 4.19: Minimal EL built successfully showed on the Hob last screen

In case that one has chosen to edit the image on the first hob screen (fig. 4.16) a new screen will appear after the packages are built and in this screen, showed on fig. 4.20, there is a possibility to add only the packages one wants to the image to build an extremely optimized or complete design, the image components choosing phase works similarly to the package choosing phase, this phase also add dependent softwares and libs when one is building one’s own system, and if one wants to remove a specific part of a software or a lib from one’s image one can remove this component from the final image.

The last step is to test the resulting images with QEMU, in order to test the images one can either use the button “test with QEMU” as showed on fig. 4.19 or by using the following command:
The QEMU screen is showed below in fig. 4.21. Notice that on the last line on fig. 4.21, the QEMU’s screen says that the images are running on a VNC server more specifically 127.0.0.1 : 5900, so virtual machine (VM) is needed to be run in order to see the outputs of the images. In order to run the VM run vinagre on the command prompt from a different terminal window.

After that the VNC server address is needed to be written and the connect button be pressed as seen in fig. 4.22 in order to be show and test the images on the virtual server, as seen on fig. 4.23
4.4.3 Deploying Minimal System

After having the images done for the BBB, the mSD card is needed to be formatted in order to make the mSD card a bootable media to the BBB, in order to do that firstly install gparted in the host system, as showed below:

```
sudo apt-get install gparted
```

In order to format the mSD card, an USB-SD card reader adapter was required because
the *Host* did not have this capability to read the mSD Card.

To call the gparted to format the SD card the command below is used.

\[
\text{gparted}
\]

In gparted two partitions were created, one formatted (and named as *rootfs*) in ext4 and one in FAT32 filesystem (named as *boot*), and then notes where the SD card was mounted were taken(sdx, where x is a number) [95].

Then the next step is to copy the kernel, bootloader and filesystem with the commands showed below [95].
cd build/tmp/deploy/images/beaglebone

uImage.bin /media/boot/MLO

cp u-boot-beaglebone.img /media/boot/u-boot.img

sudo tar -xf core-image-minimal-beaglebone.tar.bz2 -C /media/rootfs

This steps can be done in a Windows system with different formatting software or through different commands.

After that one can either boot directly from the mSD card or add these images to the flash memory.

In order to boot the BBB from the mSD card one needs to hold the boot switch for 5 seconds at the BBB, after one presses the power button [95].

In order to transfer the images to the flash memory at the BBB, the SD card should have the sdflasher, and the mSD card connected to the Host. At a Windows Host use the win32diskimager, found in this link and create a .img file with the program [96]. At a Linux Host a similar procedure can be made at a terminal with the following commands [95, 97].


xz -d BBB*.xz

sudo dd if=./BBB*.img of=/dev/sdX

And to flash the image from the mSD card to the board, the mSD card is inserted to the turned off BBB and after that the BBB's boot button is pressed and then one connects power to the board. One needs to wait the LED flashes for about 25 min, while the new image is flashed to the board, when the 4 LED are lit, the power is removed and the SD card from the board is also removed. After that the board is turned on to check if the new
image is on the board [97].

In the end of this section a diagram providing an overview of how the procedure of building an EL with Yocto Project works is showed in fig. 4.24.

Figure 4.24: Overview of the procedure of developing an EL, with two scenarios the one with solid line where a recipe is built automatically and the one with the dashed line where the EL is edited with the packages and software components the developer wants.

### 4.4.4 Defining EL requirements

For defining the requirements a Standalone Setup was used for developing the EL, as the BBB come with an Angstrom distribution installed on the board, and then the Ethernet cable was connected to download the used packages for the development of the scripts to read and send data. The Angstrom Distro already come with python and bash script languages used on the scripts in this project, and other basic components as internet software and text editors.

The internet is connected automatically if the Ethernet cable is already connected when the board is turning on, when the Ethernet cable is connected after the system is already on, the Ethernet package ethtool is needed to be called with the following command in order to connect the board to the network:

```
eth0 up
```

Despite of already having python on the Angstrom distribution, in order to control the BBB's I/Os it BBB needed a python library to control the features, in some of the Angstrom versions this library is not already installed, in order to install the BBB's I/O control library
is by using the following commands:

```bash
sudo apt-get install build-essential python-dev python-setuptools python-pip python-smbus -y

sudo pip install Adafruit_BBIO
```

If it does not work it is possible to do it alternatively with the following command:

```bash
```

Other used package that is necessary to use is the Bluetooth control one that is called `bluez`, other dependent packages are installed at the same time, like `hcitool` and `bluetooth`.

```bash
sudo apt-get install bluez
```

For log purposes BBB needs to use USB and the SD card controllers, but these controllers were already installed in Angstrom.

Table 4.7 summarizes which packages are added to the minimal EL system to build the custom EL.

<table>
<thead>
<tr>
<th>Package</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>python</td>
<td>programming script</td>
</tr>
<tr>
<td>bash</td>
<td>programming script</td>
</tr>
<tr>
<td>nano</td>
<td>text editor</td>
</tr>
<tr>
<td>ethtool</td>
<td>Ethernet controller</td>
</tr>
<tr>
<td>apt-get</td>
<td>installer</td>
</tr>
<tr>
<td>Adafruit_BBIO</td>
<td>BBB I/O controller</td>
</tr>
<tr>
<td>bluez</td>
<td>Bluetooth controller</td>
</tr>
<tr>
<td>bluetooth</td>
<td>Bluetooth controller</td>
</tr>
<tr>
<td>hcitool</td>
<td>Bluetooth tools</td>
</tr>
</tbody>
</table>
4.5 Developing scripts

4.5.1 Programming languages used

The programming languages used in this project are the bash script that has an extension of `.sh` or no extension and python which the archives use an extension of `.py`. Both of this script languages were used to program the EL in the BBB.

The ones used for the Android app are **Java** and **xml** of the same name extensions. Those were used to program the app on the Eclipse Integrated Development Environment (IDE). When the app is compiled a resulting running file has the `.apk` extension.

4.5.2 Scripts relation

The fig. 4.25 shows the relation between the scripts that are going to be described in more details on the following subsections.

In the fig. 4.25 the diagram gives an overview of how the software part of the design relates. First BBB checks if the Bluetooth is up or not on the board then if Bluetooth is up, BBB starts the Bluetooth and try to connect to the smartphone, if the connection fails the offline mode of the system runs (BBB runs the IOcontroloff.py script) with the log file saving the raw data, if the connection to the smartphone is successful BBB sends a character to the cellphone to starts the online mode of the system (BBB runs the IOcontrolon.py) that calibrates the optodes and display processed data real time from data sent to the smartphone app. The scripts are going to be divided in Init scripts, controlling scripts and processing scripts. The codes have a folder structure showed in the Appendix and the codes themselves are available at a github repository (link).
Figure 4.25: Scripts relation diagram, for both of this modes when Bluetooth is on and off, the scripts with extension of .sh are Bash scripts, the ones with .py extensions are python scripts and the one .apk is the android app.
4.5.3 Init scripts

The init scripts Onoff.sh and the Bluetootstart.sh showed in the diagram on fig. 4.25 are the init scripts responsible to start the system, look for the smartphone and either connect or not the Bluetooth if the smartphone is around, both scripts were written in the bash script language.

OnorOff.sh

This bash script is used during the system startup being on the etc folder where the init commands starts, OnorOff.sh firstly turns off the status LEDs of the BBB to avoid BBB status LED’s light to influence on the collected data, then the script verify if the Bluetooth device is on or off, if the Bluetooth is on the script calls the bluestart.sh script and consequently ready.sh . If the Bluetooth connection is off the script offline mode IOcontroloff.py is called. The OnorOff.sh command works as a service, in order to write the service the following command is used:

```
start-stop-daemon OnorOff.sh
```

The command above was used to create the service, to have this code as one startup service, the script is automatically placed on the etc folder. In the service’s base code there are already options to starts and stops the service. In this script start state, the service performs what was said on the paragraph above, when the service is in the stop state, the service kills the python scripts that are responsible for the instrument control.

bluestart.sh

This script starts the Bluetooth in the system with hcitool, open a communication channel with the cell phone. The procedures and tools used to design this script were explained by Molloy[10].
**ready.sh**

This script waits for the NIRPlotter app to be started on the smartphone, when the app is started this script send it a “x” character, before the script calls the IOControlon.py script to start the analysis.

**retry.sh**

This Script is the one to restart the USB dongle when Bluetooth fails to start during boot. retry.sh is called by the OnorOff.sh script before starting the Offline mode.

### 4.5.4 Controlling scripts

The controlling scripts IOcontrolon.py and IOcontroloff.py are responsible to control when each LED is activated and at which frequency the LEDs are activated, and these scripts also read and process the data received by the detectors. The difference of which of this scripts is called is if the smartphone Bluetooth is connected or not. The summary of the controlling scripts functions are showed on fig. 4.26.

**IOcontroloff.py**

This python script is responsible to control the digital outputs that activate the LEDs alternately, read the electrical data from the detectors and write the raw data log, which the data of the two detectors when the 765nm is on, when the 850nm LED is on and the time of between readings is 0.2s (5Hz device), the resulting value of the readings from the analog inputs is a value between 0 and 1 that is used as OD, this script overview is showed on the fig. 4.26 (b).
IOcontrolon.py

Besides doing the same functions as the IOcontroloff.py script, this script also calibrates the system by reading the data during approximately 30 seconds then the script creates an averaging filter on the following readings. IOcontrolon.py also uses the MBLL to calculate the [Hb] and [HbO₂]. The averaging filter were called through “def” functions on the python scripts, inspired by the m-files MBLL.m and refine.m. And lastly this script calls the writer.sh script to send data to the smartphone.

writer.sh

This bash script sends the [Hb] and [HbO₂] data via Bluetooth using the following serial protocol: #[Hb] + [HbO₂] + time(s) + * it was used because the Bluetooth control were all out of the python script.

4.5.5 Processing Scripts

These scripts are the ones used to process the rawdata on a Desktop machine and the ones used on raw data tests to develop processing features on the controlling scripts.

Retrieving raw data

In order to get the data from the BBB, a pendrive should be inputted to the USB port and the FTDI cable to the BBB. When the system is started firstly turnoff the IOcontroloff.py to stop the log file by using the following command:

```
killall python
```

After doing this step, the port where the pendrive is added is checked by using the command below.
Figure 4.26: Controlling scripts diagrams’ two modes. (a) Online mode diagram. (b) Offline mode diagram

\texttt{fdisk -l}

The pendrive is going to have a pathname as \texttt{\dev\sdax} where “x” is going to be a number, knowing this number one can mount this drive into a BBB’s folder, like \texttt{\mnt}, this mounting step can be done by using the following command:

\texttt{mount \dev\sdax\mnt}

Then with the file mounted, one can copy the datalog file as a .csv file on the drive just
mounted.

```
cp datalog \mnt\data1.csv
```

After copying this file one can either make the datalog empty, or unmount the pendrive by using the following commands respectively:

```
cp \dev\null datalog
umount \mnt
```

m-file processing scripts

The m-files scripts were used both to process the raw data and to serve as basis to write the processing part of the IOcontrolon.py script that shows real time data, observing the code on the git repository, the control scripts use `def` functions in python to call analog functions that were previously tested and written in MATLAB.

Regarding the raw data processing the first script to be used is `rawdataread.m` that splits the data on the logfile into 4 resulting arrays referring to the 2 PDs in the 765nm and 850nm, in order to do this, the cells one wants to be on one’s analysis needed to be known, so a look on data in a spreadsheet software like Microsoft Excel in order to define the desired interval of data to be analyze before running the command is required. Table 4.8 shows an example of how the raw data was saved in the logfile with a comma separated value format.

<table>
<thead>
<tr>
<th>timer(s)</th>
<th>pd1</th>
<th>pd2</th>
<th>NIRLED</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.855</td>
<td>0.679</td>
<td>850nm</td>
</tr>
<tr>
<td>0.2</td>
<td>0.840</td>
<td>0.2764</td>
<td>765nm</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Then these arrays are used with the `refine.m` function to use the averaging filter with
the whole set of data. Then the resulting array passes through a low-pass 4 Hz butterworth filter using the `lowpassbutterworth.m` function. The filtered data then is used as an input to the `MBLL.m` function to calculate both the [Hb] and [HbO₂] by using the eq. 2.4 and eq. 2.5, the distance values showed on table 4.9, and the coefficient values showed on tables 4.10 and 4.11. The output arrays then can be used to plot the variation of [Hb] and [HbO₂] vs. time.

### 4.5.6 Testing the scripts

In order to test the Bluetooth functions of connecting, sending and reading characters on the BBB with a smartphone an app was downloaded for this purpose, that is a chatting app for BBB with an Android cell phone called EBB Bluetooth [10], the screen of this app is showed on fig. 4.27

![Figure 4.27: App installed for testing Bluetooth communication](image)
4.6 Evaluating raw data

The first tests were performed in the prototype by using an automatic cuff in the right arm with the prototype touching the application part on it as showed on the fig 4.28, in a dark room using the offline mode of the prototype, the test was performed twice in a 2 minute-interval. From these data noise removal techniques were evaluated, the effect of MBLL algorithm on the signal and data adjustments on a MATLAB program. After these tests the needed codes to treat data were designed for python.

![Figure 4.28: Setup to evaluate raw data obtained](image)

When first analyzing the data an averaging signal method is used to make the signal less noisy as showed on fig. 4.29, this method consists in averaging the data of the signal and then normalizes the data by subtracting the averaged data by the standard deviation of original data, subtracting those points higher than the average and summing the standard deviation to those points smaller than the average. There was a code used in MATLAB to do this procedure called refine.m (available on the github repository).

The next step then was to analyze the frequency spectrum of the data, in order to do that the power spectrum was created in MATLAB through the m-file furie.m in the
Appendix, so the averaging technique was possible to show itself as effective to reduce the signal noise. The frequency spectrum of the raw signal is showed in fig. 4.30 (a) together with the one of treated data in fig. 4.30 (b), and the data getting less noise is possible to be observed by removing frequencies around 4 and 6 Hz at the treated data.

By observing fig. 4.30 (b) there was still some noise left that could be gotten rid of,
so with the spectra’s observation a 4Hz second-order low-pass Butterworth filter could be used to minimize the noise and the resulting signal is showed at fig. 4.31, the code lpfButterworth.m was used to filter this noise (this m-file is available on the github repository).

Figure 4.31: Filtered sample raw data with a 4Hz second-order low-pass butterworth filter

Comparing the graph at fig. 4.31 with the fig. 4.29 the noise was substantially reduced by using these two techniques (Filtering and averaging filter).

4.7 Applying MBLL to data

In order to apply the MBLL the Equations 2.2, 2.4 and 2.5, and the same set of data from the previous section, with the addition of data from the 765nm source were used. The distances among the sources and the detectors were required to be known, these values are showed on fig. 4.9. The values of the constants contained in these equations were also researched.

<table>
<thead>
<tr>
<th>Wavelength(nm)</th>
<th>PD1(cm)</th>
<th>PD2(cm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>765</td>
<td>2</td>
<td>2.3</td>
</tr>
<tr>
<td>850</td>
<td>2.3</td>
<td>2</td>
</tr>
</tbody>
</table>

In table 4.10 the extinction coefficients used are showed, these coefficients refer to
study [98], they did not had the values for the wavelengths used (765nm and 850nm) so a linear interpolation was made for these values.

<table>
<thead>
<tr>
<th>Wavelength (nm)</th>
<th>$\epsilon_{Hb}$ ($mM^{-1}.cm^{-1}$)</th>
<th>$\epsilon_{HbO_2}$ ($mM^{-1}.cm^{-1}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>765</td>
<td>0.335</td>
<td>0.159</td>
</tr>
<tr>
<td>850</td>
<td>0.191</td>
<td>0.2764</td>
</tr>
</tbody>
</table>

Regarding the DPF, the study used was [99], in this study there was not the wavelengths used in the experiment either, so a linear interpolation was used to acquire the approximated DPF values, the resultant values are showed on table 4.11.

<table>
<thead>
<tr>
<th>Wavelength (nm)</th>
<th>DPF</th>
</tr>
</thead>
<tbody>
<tr>
<td>765</td>
<td>6.45</td>
</tr>
<tr>
<td>850</td>
<td>6.25</td>
</tr>
</tbody>
</table>

The resulting graph from the test of the application of the formula to data is showed on fig. 4.32. In this figure one can observe that when there is a decay of the $[HbO_2]$ the $[Hb]$ is increased, these regions represent when the cuff starts tightening the arm, and when the arm is released the $[HbO_2]$ increases again while the $[Hb]$ decays, and after some seconds the hemoglobin concentrations come back to a similar pattern at the start of the graph. A good contrast is shown between $[Hb]$ and $[HbO_2]$, unless between 60-80s that can be seen some crosstalk. There is also a m-file that was used to do this called MBLL.m available on the github repository.

### 4.8 Developing Android GUI

The codes used to develop the GUI can be found on this link and how the codes are structured can be observed on the Appendix section.
4.8.1 Installing Eclipse to Android Programming

By default Eclipse is just a Java Programmer, in order to use it to deploy the created Java programs in an Android phone the Android Development Tools (ADT) plugin is needed.

Eclipse is available to Windows and Linux OS, during the GUI development a Windows machine was used. The Eclipse version Luna IDE for Java Developers was used [100]. When the file was completely downloaded, the file was unzipped and Eclipse was ready to go.

The next step was to add the ADT plugin to start the app development. In order to activate this plugin the following actions were done:

1. The Eclipse was started
2. In the menu “Help”, “Install new software” was clicked
3. When a new window appeared the button “Add” was clicked
4. On the next window, in the repository box, the “ADT Plugin” was inputted on the
“Name” field, and at the “Location” field “https://dl-ssl.google.com/android/eclipse/” link was inputted

5. “Ok” was clicked then

6. On a checkbox screen the ADT plugin was selected

7. “Next” was clicked

8. The downloaded tools are showed, “next” was clicked again

9. A “License and Agreement Screen” was showed, the “Finish” was clicked at last

10. After ADT plugin was successfully downloaded Eclipse is required to be restarted [101].

After downloading the ADT the location of this just downloaded tools needs to be specified. In the “Welcome to Android Development Window”, one has to click “use existing SDKs”, and then one has to browse the folder of the just downloaded packages then click next. And then Eclipse was ready to program Android apps [101].

4.8.2 Setting the Android phone to be able to debug the apps

Most of steps needed to enable Debugging apps designed in Eclipse through a cell phone, is on the cell phone itself. Firstly one needs to enable USB debugging in the device, in order to do this:

1. On an Android phone go into cell phone “settings”, then “about phone”

2. Click 7 times the “build number” to enable “developer options” on the phone,

3. In this tab select the “Enable debugging through USB” on the phone.
4. When connecting the cell phone via USB to the computer the “Debug option” is now available on the cell phone [102].

When compiling an app, Eclipse now will be able to debug the software on the phone.

4.8.3 Developing a minimal System

Creating an Android Project with Eclipse

The Android version “Jelly Bean 4.2” was used to develop the system, the Eclipse with the Android ADT can develop from 1.0 to 5.2 versions.

In order to create an Android project the first thing that happens when the eclipse is opened Eclipse requires the workspace to be chosen. After that to create an Android project one needs to follow the procedure below:

1. Access “File” > New... > Project...

2. Select “Android Application Project”, click next

3. Name the project and select the version of the Android Cell Phone

4. Leave “Create Activity” and “Create Icon” selected, then click “Next”

5. Create an icon

6. Choose the type of the template as “create a blank activity”

7. Click “Finish” and then start programming.

Eclipse automatically creates .xml files and .java files that have the essential code of an app. If the default names on project creation are used the files that are the ones needed to be edit are:

- **MainActivity.java** - Located at the “src” folder, This is the main file that is going to run first on the device.
• **activity_main.xml** - Located at the “res/layout” folder, this xml file is responsible to the layout of the interface screen, this file can be edited in a xml form mode, a text mode, or in a graphical layout interface.

• **AndroidManifest.xml** - Located at the core folder of the project this file contains permissions (like the Bluetooth ones, that must be use in order to use the smartphone Bluetooth), version control, and other essential information to the file interact with the Android.

The procedure of compiling, running and testing the program is the same, one can either compile the program and not test the app and/or compile the program, run and debug the app in an Android smartphone or smartphone emulator. In order to compile/run/debug the app the developper needs to press the Run button or activate the compilation by the Run Menu >Run.

If the cell phone is not connected a .apk file that appears on the bin folder at in the project chosen workspace can be used to install this app on the cell phone. In order to transfer the file to the cell phone, assure that unknown sources installing is enabled on the phone (at the menu/ settings/security) and then one can install the .apk by clicking the file. If the smartphone is connected to the computer the file can be installed directly on the phone, and the app will be started and debugging the app on the Eclipse’s LogCat is possible (fig. 4.33).

![Eclipse's LogCat debugger, debugging an app showing the cellphone processes messages](image)

Figure 4.33: Eclipse's LogCat debugger, debugging an app showing the cellphone processes messages
**Communication protocol between BBB and the cell phone**

The minimal system to be used in this development was based on an application that firstly find and connect to a Bluetooth device and then read a protocol of serial data and display in a text format, the basis for this application can be found on the following sources [103, 104], the code can be found on the github with the name of “NIRDateread”, and indicated on the Appendix.

As said on the last subsection in order to communicate to a Bluetooth device, the permissions should be set, in order to do that the following two lines were added to the AndroidManifest.xml:

```xml
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
```

Other important component for programming Bluetooth is by importing bluetooth. BluetoothAdapter and BluetoothDevice, were used to connect the bluetooth from the cellphone automatically to the BBB. On the MainActivity.java file some important data is assigned as the device (BBB) address (“00:1A:7D:DA:71:13”) and the UUID used for Serial communication (“00001101-0000-1000-8000-00805F9B34FB”), besides of this, there was another important thing in this design, the program only starts on the BBB after receiving a character sent by the smartphone that was arbitrarily chosen as “x”. If more details to the Bluetooth communication is needed the codes can be seen at the github, and the reference [103] guided this design.

Finally talking about the communication itself, the serial protocol used was based on the application [104], and the protocol work in the following way:

#1.23 + 2.34 + 3.23 + 2.43 + *

Where # is the character that indicates the data string to be read, + is the one delimiting each value to split data to the desired values and the * character was the indicator that
the data string was over, this protocol idea was used on the BBB to send the data to the phone, a minimal application that uses this protocol was adapted from [103, 104] and this minimal app is showed on fig. 4.34.

![Image](image.png)

Figure 4.34: Minimal system app to receive data from BBB via Bluetooth, using the application NIRdataread

One difference on the designed minimal app showed on fig. 4.34 is that in the design proposed a function called StringTokenizer is used with the delimiter + which allows the data delimited can have a larger number of characters on each delimited the data using large numbers or very small numbers, differently from the base app [104] which the data was delimited by the number of characters.

### 4.8.4 Developing the Final GUI

The GUI development also had another app as basis [105], availble on github repository with the name of “Plot1” as indicated on the Appendix. A java library was used for designing graphs for Android applications called Achartengine, that can be obtained on this link, firstly the developer needs to add this library to the app, in order to do add this library:
1. Download the library

2. Move the obtained .jar file to the lib folder

3. Right-click the file and on the build path option

4. Choose the “add to build path” function

The example given in [105] is showed in fig. 4.35. In this example several tools in the final design used this app as basis, the use of point and line, the edition of the color, size of text, legend, and other parameters that can make the graph with the desired layout.

![Figure 4.35: Achartengine example app with random data, using the “Plot1” application](image)

After using the base files, the layout was corrected by going to MainActivity.Java and on the Graphical layout tab, and the Landscape view was chosen besides the layout items from the minimal software. The final version of the built application called NIRPlotter is showed on fig. 4.36. Besides the graphical part features as moving the graph by sliding one’s finger and by using the zoom tools on the bottom right of the app were added to the final app.
4.9 Validating the System

4.9.1 Battery Testing

Tests to evaluate how long can the battery stand were made, the first scenario was with the device with a full-charged battery in the offline mode (with the bluetooth module not connected). The second scenario was also with a full-charged battery but this time an online mode (with the bluetooth module connected and the cellphone) showing the real-time data.

Another testing realized was how long it take the battery to get full-charged and what was the maximum voltage the battery got at full-charge.

4.9.2 Occlusion Test

The occlusion test realized to validate the system was made with an automatic pressure cuff showed at fig. 4.28. In a well illuminated room, the pressure cuff was worn on the left arm of the subject, then the device (with full-charged battery) was turned on, after the
device was completely on, at the smartphone with Bluetooth on the NIRPlotter app was started, then the device was placed facing down the left forearm of the subject. After 30 seconds of calibration the data starts appearing on the smartphone screen, if data is too low or noisy the device is slightly moved to acquire data at other position. The protocol followed was to keep the subject steady for 15 seconds then the automatic pressure cuff is turned on, then when the cuff releases the arm (40-60s later), a 30-second relaxing time wait was given to the subject then the pressure cuff was turned on again. The occlusion happened 3 times, the times when the occlusion starts and ceases were written down then used as the occlusion time intervals. When the analysis finishes a “printscren” command was used on the smartphone to save the graphs, and the logfile was acquired from the BBB as well.

Then the experiment was repeated in a dark room.
Chapter 5 - Results

In this section the more relevant results regarding the designed prototype and the device’s validation are showed.

5.1 System specifications

Mechanical Specs

On the following table 5.1, the mechanical specifications of the design with and without the rubber case are showed.

Table 5.1: Mechanical specifications of the design with and without the rubber case. [99]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>No Rubber Case</th>
<th>Rubber Case</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight (g)</td>
<td>95.34</td>
<td>195.22</td>
</tr>
<tr>
<td>Dimensions (mm)</td>
<td>30x59x87</td>
<td>43x77x92</td>
</tr>
</tbody>
</table>

The components that mostly contribute to the device’s weight are the rubber case (≈100g) the battery (≈50g) and the BBB (≈40g). Regarding the dimensions the biggest contributor to is the BBB (9x5.5x1.5cm) to this size. One important factor to stand out is that this weight and size of this device are relatively small (comparing it to a commercial similar devices like the ones showed on table 2.4 and device could be used as a wearable one with the potential to not bother the subjects during analysis, and do not impair the subject’s
movement during functional analysis. Other important thing is that the device being built is a prototype, so by using a 3D modeled case, a PCB instead of a handmade soldered board, and an edited BBB (without unnecessary features) the device can potentially reduce the its size and weight to even lower values.

### Battery Specs

The following table 5.2 shows the results from the battery tests referring to the battery specs.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power (mAh)</td>
<td>2000</td>
</tr>
<tr>
<td>Max Voltage (V)</td>
<td>4.25</td>
</tr>
<tr>
<td>Min Working Voltage (V)</td>
<td>3.7</td>
</tr>
<tr>
<td>Duration Time (h)</td>
<td>4h</td>
</tr>
<tr>
<td>Recharging time with 50mA current (h)</td>
<td>2.5h</td>
</tr>
<tr>
<td>Battery Life (y)</td>
<td>3</td>
</tr>
</tbody>
</table>

The system work with a smartphone on the online mode, so there are cases that the battery on the device could lasts more than the smartphone, the device will keep on sending the Bluetooth data until its battery cannot stand for too long, but the program can crash if the Bluetooth communication between the smartphone and the BBB gets down due the phone out of battery.

### Cost

The cost of every single item can be observed on tables 4.1 and 4.2 and the total cost of the project is US$ 143.60. Only considering the items that are components of the design the cost is US$ 105.20, for a single device, in a large scale production with retail prices the board could potentially reduce the device’s price. The other US$ 38.40 represents the
items used only during development so these items will not add extra cost to more devices being built with this idea.

As a matter of comparison to a commercial device, according to Scholkmann’s Review [27], NIRS devices prices ranges from approximately US$ 10,000 to US$ 100,000, the ones cheaper are mostly the ones that uses LD or LEDs sources that can either be wireless or not. The ones that are the most expensive systems use a lot of channels or Ultra-fast lasers as sources, several optical fiber cables and PMTs that makes the system more complex and expensive. And according to Table 2.4 the price for commercial wireless NIRS systems can range from US$ 3,000 to US$ 33,000 and there should be a relation to the number of channels the devices have and the use of LDs, because the ones more expensive both have LDs as sources and have 16-32 channels.

Even tough considering additional regulatory costs, labor costs among other production costs this idea of device can substantially reduce the cost of low complexity devices.

5.2 Final Chronogram

On fig. 5.1 the chronogram adjusted after all activities accompanied to be done are showed.

The project delayed two months, and comparing the final chronogram to the one in Fig. 4.5. One can observe that the “Design the GUI”, “electrical tests”, “software tests” and “Linux System Development” tasks have not delayed, and the two parts that most delayed were the “Plan BBB Scripts”, “Validate Design” and “Writing”, these tasks might have delayed because these tasks needed to be done iteratively, while the other activities were going on this activities had to change due to practical issues or plan changes.
5.3 Occlusion Test Results

5.3.1 Online device in a lit scenario

On fig. 5.2 we can observe the results obtained for the occlusion test with the device in the device’s online mode, tested in a lit room.

Observing the occlusion intervals, that occurred 3 times during the validation procedure showed by fig. 5.2, when occlusion starts the [HbO₂] starts decaying while the [Hb] starts increasing this represents the reduction of oxygenation when the tissue is occluded as expected, and when the arms are released the opposite happens [HbO₂] increases and the [Hb] decays representing the moment when the blood flow starts coming back to normal and consequently make the oxygenation back to normal. The signal still seems noisy, the signal could be improved with a more complex filter on the signal. The signal around the 850s represents the moment when the device was being removed from the patient, and detecting only environment light data.
Figure 5.2: NIRPlotter showing the online scenario results for the occlusion test in a lit room. The intervals within an indicating “white O” represents when the pressure cuff started to occlude the arm.

5.3.2 Logfile lit scenario

A logfile was created while the experiment showed on fig. 5.2 was running, this logfile was processed in MATLAB, using the averaging signal technique of the whole data (not only the 30s calibration moment), and a low-pass second-order 4Hz filter. The resulting wavelengths are showed on fig. 5.3.

The peaks seen on fig. 5.3 matches to the ones seen on the 5.2, however these peaks differ on the other regions presenting less noise, and higher magnitude peaks of the concentration during occlusion, this difference on the amplitudes could have happened due the whole data treatment for calibration, and the second-order filter can both give a higher contrast between the curves but this filter can also filters the smoother pattern of the increase and decrease of the concentration curves on the fig. 5.2.
Figure 5.3: NIRPlotter showing the offline scenario results for the occlusion test in a lit room. The intervals within an indicating “black O” represents when the pressure cuff occluded the arm. This signal has been filtered by an averaging signal of the whole data and by a low-pass second-order 4Hz Butterworth filter.

For both results one should remember that an automatic pressure cuff was being used instead of an regular one, so the cuff could not be controlled on how long the cuff would keep the arm occluded, and the cuff usually could not keep the arm occluded as long as in done in other protocols [2] that kept the arm occluded for 2 minutes before the arm was released.

### 5.3.3 Online device in a dark scenario

On fig. 5.4 we can observe the results obtained for the occlusion test with the device in online mode, tested in a dark room.

The fig. 5.4 curves do not differ by much to the ones observed on the fig. 5.2, what we can drag an interpretation that the rubber case really helps on the isolation of environment light, showing that this device could be a good option to outdoor applications where we could not control the light as the validation tests done for this prototype.
Figure 5.4: NIRPlotter showing the online scenario results for the occlusion test in a lit room. The intervals within an indicating “white O” represents when the pressure cuff occluded the arm.

5.3.4 Logfile dark scenario

Another logfile was created while the experiment showed on fig. 5.4 was running, this logfile was processed in MATLAB, using the averaging signal technique, and a low-pass second-order 4Hz filter. The resulting wavelengths are showed on fig. 5.5. The result obtained on the fig. 5.5 compared to the 5.3 supports also the idea that the environment light does not add considerable noise to the system, due the use of the rubber case to reduce incoming light to the detectors. The contrast on the magnitude of the concentrations during occlusion persists in this test, The evaluation of which filter is better is needed regarding the time for calibration for both of them to keep the important data patterns of the increasing/deacreasing of the concentration curves (as seen on the figs. 5.2 and 5.4) and keep the concentration curves’ contrast at peaks (as showed on the figures 5.3 and 5.4).
Figure 5.5: NIRPlotter showing the offline scenario results for the occlusion test in a lit room. The intervals within an indicating “black O” represents when the pressure cuff occluded the arm. Using the averaging signal technique on the whole data, and a low-pass second-order 4Hz.
Chapter 6 - Conclusion

In this Section comments on some situations that happened during development and about the results gotten, conclude this project and discuss some ideas for future work will be provided.

6.1 Discussion

Cost reduction

The board cost could be even more reduced with the release of the BeagleBone Green that is a version of the BBB without the HDMI connector and graphical capabilities, the move of the USB host to the other side of the board and the addition of Grove connectors in the free space, this version of the BeagleBone was launched this Aug/2015, and costs US$ 39 (US$ 16 less than the BBB used in this thesis), what could reduce even more the price of the project, and the functionalities being used of the BBB in this prototype are still being part of this new version of the BeagleBone [106, 107].

Besides potential reduce in price of this prototype, the device is considerably cheaper than similar commercial devices, comparing the NIRS commercial devices’ price on Table 2.4 that ranged from US$ 3,000-33,000, and the Scholkmann et al review’s [27] where prices ranges from US$ 10,000-100,000, even though if one considered that the proposed design only has 2 sources and detectors there is no regulatory or labor associated costs,
this prototype has the potential to be a potentially cheap device since the development expenses were US$ 143.60 for a single device, that low-cost device could make feasible to applications like home care and use of this device to poor areas as sought by Zhang et al[22].

**Designing time reduction using EL**

This is a relative factor that could not be taken into account because there is no comparing data to define if the EL really reduced the development time, However by observing fig. 4.5 and 5.1 one can verify that the EL development was not the major responsible to this project duration.

By running studies of trying to develop the same or similar system with different approaches, and by designing a new system use the one as base one could verify if development time is saved.

**Signal artifacts**

Movement artifacts were only perceived when the application part of the device was fully removed from the application to the skin, when it was fixed or hold the movement artifacts were not an issue at all.

**Battery**

Before trying to use a LiPol 2000mAh battery a LiPol 1200mAh one was being used but this lower power battery could not support the required power to start all items connected to the board at boot, one could observe that a voltage peak at the board start was required to make the Bluetooth dongle work on start-up. This battery added more working time and a longer time to recharge the battery.
**Bluetooth dongle Start-up issue**

The problem of Bluetooth failing sometimes to start persisted, so a script to re-initialize the Bluetooth dongle was created in case the dongle does not start based on this thread [108]. The correcting script is showed on the git repository and in the thread [108], this code is on the git repository named as “retry.sh”.

**PCB Design Consideration**

In order to reduce size, weight, production time, a commercial version of the device could use a PCB for the hardware on further development, a PCB also has potential to reduce price in the long run, in a possible manufacture of several devices like this. and it also could reduce the size and weight slightly of the design, by using a perfectly fitted board and by not using too many jumpers and solder as the handmade cape.

**Sources and detectors issues**

In this design the idea was to use LEDs as sources, because LEDs were advantageous due their price, however LEDs comes with a wider light bandwidth and noise associated with the signals, and LEDs’ capability to go deeper into tissue are not so strong. One suggestion could be used are the LDs in place of the LEDs, that would come with a higher power and a narrower bandwidth to increase the devices'light detection accuracy, by not increasing the device’s price by much.

Another downside to the proposed design was the use of fixed sources and detectors, this problem could be improved by the use of flat cables on the sources and detectors application parts in order to have some flexibility on making up different protocols as the following studies [16, 21]. Or as the low price is a good size of this device it could have different removable capes with different protocols, extending the options this device could have.
There was a limitation regarding the use of *BBIO_Adafruit* python library to read the data, because the library’s reading function needs to read the data twice by using the `ADC.read()` command twice as said by [109] because this function needs a second read to clear the buffer, before receiving a new data.

Sometimes getting a coherent signal was not easy so during the tests the application part was moved to other place on the subject's arm and press the board toward the subject's skin, or the sources could be placed closer to the from detectors in order to get a better signal.

**Possible improvements on the design**

One discussed thing was that when applying the MBLL in the algorithm, the prototype user could have the DPF as an input since DPF is a different constant from baby to adult, female to male, body part to body part, so this extra feature would be an interesting for the user in a future version of the project. Continuing talking about the GUI functions to save the data on the smartphone and not only displaying it, and gathering the graph by using a “printscreen” function on the smartphone could be added. On a further version with more channels a tool to navigate between channels to observe data on a specific channel and an overview of all channels would be needed. There is an advantage of having no cost on designing an Android app, among the health professional the use of IPhones are more popular according to Ventola [25], (80% uses IPhones as their smartphones’ preference) so an IPhone version of this app could be widely accepted by healthcare professionals.

The case was functional but the case doubled the device’s weight, and due the rubber’s flexibility sometimes it moved the sources and detectors from their places making some noise or bad connection of the sources and detectors to the subject’s skin, so the device should be handled carefully. The cape was too thick and sometimes the cape would also shield the back-scattered photons making the analysis of data, null or almost null, so even the positioning should be really good planned. One alternative to the problems would be
creating a 3D modeled plastic case so this case could be exactly fit on the prototype. But the idea of the rubber application part is a good one, because the rubber offers more comfort to the subjects in long-run experiments, the comfort provided could reduce the subject’s distraction, so this rubber application could be kept on the design with the addition of a lower weight case.

Some random connectors were used as a battery connector but there is a suitable JST female connector one indicated by [91], the name of the connector is S2B-ph-K-S(LF)(SN), and this connector cost about US$ 0.16 one unit on the digikey.com portal. It would make the battery fit perfectly, and be more stable in the design.

Another battery improvements would be to make the device rechargeable on board, and notify the user when the Bluetooth will stop working, or the entire device, or prevent the device to turn off due low battery on an unsafe mode. The battery charge could be measured directly on board by using the I2C [91].

In this design 2 out of 8 possible analog inputs on the BBB were used, so more inputs to more PDs could be used, or other physiological data inputs like electromyography, electroencephalogram like Guo et al did in their design [63] could be used, among other data that could be sent in those remaining analog inputs.

A combination of two factors could be really useful on future projects that is the low-cost nature of this project and the modular nature of the Linux and as the NIRS has different configurations of source-detectors, different modalities. A good feature would be to use the same system running an EL that could be able to work with several specific purpose application part, for example DOT breast imager, NIRS for the frontal lobe of infants, and each of these specific purpose application part could use different capes designed as the one designed for this project.

There is also the possibility of using other more advanced development boards that uses Linux such as redpitaya [110] that can acquire data at a much faster rate (125Ms/s) and in a higher voltage than the BBB, making it possible to use the created EL as basis to
Some issues on the device

Sometimes keeping the device steady was hard, so making the device wearable would help to make it fixed and reduce artifacts due misplacement of the optodes.

The calibration part of the online mode should be improved, both for the Butterworth filter and for the averaging filter, because in the validation results the used Butterworth filter removed the smooth increasing/decreasing curves of [Hb] and [HbO₂] on the offline analysis on fig. 5.3 and 5.5, as the contrast between the [Hb] and [HbO₂] not seem so clearly as the offline mode on fig. 5.2 and 5.4, it should be met a combination of this two filters where we do not lose these features of each mode off the device.

6.2 Final Considerations

This thesis showed the building and the validation procedure of the design and validation of a low-cost wireless NIRS using EL with a smartphone GUI, the device’s price (US$ 143) despite being an unique prototype, was relatively lower than other commercial NIRS devices (US$ 10,000 - US$ 100,000), and wireless NIRS devices (US$ 3,000 - US$ 33,000) this low-cost feature is an important trait when thinking of home care devices and devices intended for low-income regions.

The device’s low-weight and small-sized features comparing to other devices, were also important considering the device has potential to be a wearable, broading the possibilities of application to function analysis without distracting the subjects.

The real-time interface on an Android phone is something not found in literature yet for this kind of device, and this feature could help on future telemedicine, mobile health care and home care applications.

The system was successfully validated but the validation procedure could use a reg-
ular pressure cuff to make the arm stands for two minutes occluded or more then clearly see the contrast between the [Hb] and the [HbO₂], and compare results to other literature prototypes using similar tests of validation. For this system a review in the calibration and filtering procedure would be necessary for both modes in order to not miss important data when displaying the data on the smartphone or in MATLAB, and to reduce noise.

The use of EL was important to show the usefulness when making up a project providing a lot of options, and providing some ready tools like the Bluetooth libraries and applications, making the design faster and kept a low-cost device. The EL in this design was low-sized and precise for the application that we needed.

6.3 Future Work

Improving the device’s complexity, making the device able to use more complex applications such as FD systems, SRS systems, by using LD sources instead of LEDs, and more PDs are planned. Making the system wearable is also intended to make a wearable system in order to keep the optodes more stable. Further studies with this device intends to perform functional analysis of subjects to evaluate the advantage of having a small, light device on someone's head or muscle, having a shorter movement artifact, more comfort to subjects and less distraction to dragged by the NIRS device.

Some relatively quicker adjustments on the prototype that were proposed on the discussion section like using a 3D modeled case, a PCB design of the cape, the development of the similar app for IPhone since IPhones are more popular among health care professionals are also planned to be done.

The EL was of great help in this design, but for further designs the created EL could be even more essential since the base code that was made for this design could be extended to a lot of other application, with a larger number of sources and detectors. Having this base EL could improve the speed of new application software are designed. One important thing
that needs to be stood out is the fact that NIRS is a field with a wide range of applications, different modalities, different chromophores different scenarios, different source/detector configurations, and having a versatile system that could be edited to each case could help on the development of a complete NIRS system, specific task designed devices, or even devices that could be integrated to other ones. The use of EL in similar projects could also verify if the use of EL during the design really improves the speed of development.
Bibliography


[99] L. Gao, C. E. Elwell, M. Kohl-Bareis, M. Gramer, C. E. Cooper, T. S. Leung and I. Tachtsidis. “Effects of assuming constant optical scattering on haemoglobin con-
centration measurements using NIRS during a valsalva manoeuvre”. *Oxygen Transport to Tissue XXXII*, 2011.


Appendix

Git

The files were uploaded at the github repository, they both can be observed and downloaded under the idea of the GPL. The link to access the files follows here. In this link enter the Desktop folder and find three folders which contents are going to be shown on the next set of pictures.

Github repository main folder
App files

Android app projects used during development

Contents on app folder used during development

Contents on the NIRPlotter folder
Graph and Bluetooth classes files used on the app, contained on a src folder

BBB scripts

BBB folder containing Linux system folders with the scripts that control the BBB system
Contents on the root folder

**MATLAB offline m-files**

MATLAB contents. The files used to process raw data.