POLITECNICO DI TORINO

Master Degree Course in Electronic Engineering

Master Degree Thesis

Design and Development of a Multi-Parameter Wearable Medical Device

ECG and PPG Watch

Advisors
prof. Eros Pasero
Eng. Jacopo Ferretti
Eng. Vincenzo Randazzo

Candidates
Domenico Motta

Academic Year 2019-2020

Summary

Nowadays it has become very important to monitor some vital parameters through the use of wearable devices and to save these data on external devices such as mobile phones or tablets.

The purpose of this thesis is to study and create a wearable instrument similar to a bracelet able to monitor the electrocardiogram (ECG) through a one lead measurement and the blood saturation (SpO_2) through an integrated optical module given by MaximIntegrated^{TM}. All data sampled by the device, can be exchanged via Bluetooth® on a mobile App to have the possibility to process them and present results to the user in an easy way in order to facilitate their reading.

The device consists in two PCBs one for the ECG monitor and the other one for the SpO_2 measure; they are connected together via flat cable so as to insert them in a single case. This system is based on a Texas Instrument 32-bit microcontroller, programmed in a RTOS way. The advantage of Real Time Operating System is the multitasking operation, different tasks in the same project can be easily synchronised.

The project starts from an old version of the device and consist in design a new analog front-end for the ECG, choose and insert the blood saturation sensor, draw the new schematic, route all the component in the PCBs, mounting the board, program the microcontroller, update the Android^{TM} App and finally test for correct operation.

To sum up, this system is used to provide hearth rate, anomalies such as atrial fibrillation and blood saturation. There are several future perspective which consist in introduce a memory inside the board, some other health sensors and adapt the project to more specific purposes such as calculate blood pressure in a non-invasive way through an innovative algorithm that is being developed.

Acknowledgements

Ringrazio il Prof. Eros Pasero per la disponibilità dimostrata nei miei confronti e per la possibilità che mi ha dato di lavorare a questo progetto. In questi mesi ho messo in pratica le mie conoscenze apprese in questi anni di studio e ho acquisito nuove competenze imparando ad usare nuovi tool per la progettazione di PCB e per la programmazione di microcontrollori.

Un ringraziamento speciale va anche a Jacopo che mi ha seguito durante questa tesi, rendendosi sempre disponibile nel momento del bisogno, il suo aiuto è stato molto prezioso per il raggiungimento del risultato finale. Un grazie speciale va anche a Vincenzo per il supporto fornitomi per quanto riguarda lo sviluppo dell'App.

Vorrei ringraziare i miei genitori che mi sono stati al fianco in questo percorso scolastico consigliandomi nelle scelte e sorreggendomi nelle difficoltà. Un grazie speciale va a mia sorella Chiara e a Riccardo sempre presenti nei momenti di necessità. Un ultimo grazie va a tutto il resto della mia famiglia, nonni e zii.

Contents

List of Tables

List of Figures

Part I

Introduction

Chapter 1

Introduction to the problem

The ECG sensor is a tool widely used in the medical field for the measurement of contractions of the muscle tissues of the heart during heartbeat, through the application of electrodes on the skin. The meaning of ECG is electrocardiogram and the result of this acquisition is a signal/waveform, Figure 1, sum of every single wave generated by every single movement of the heart Figure 1.

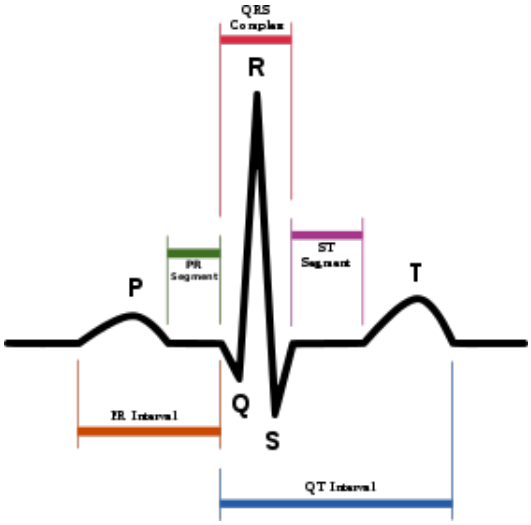


Figure 1.1. ECG waveform.

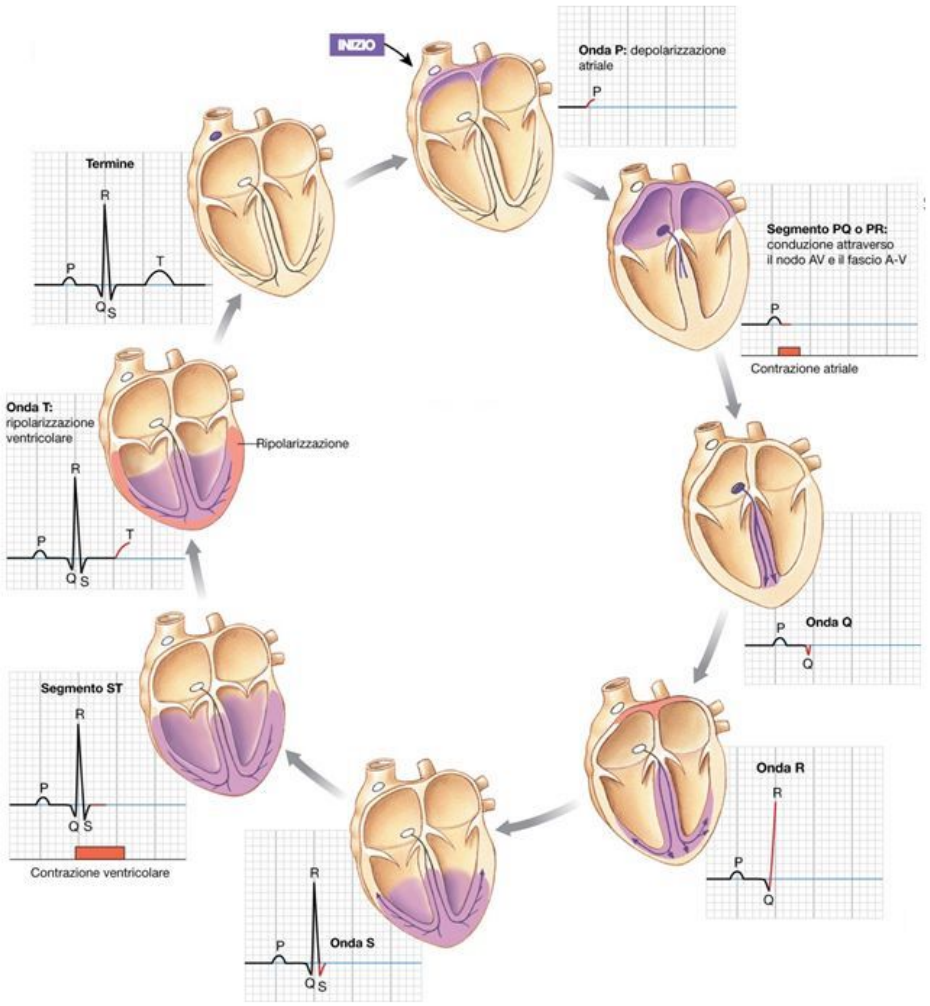


Figure 1.2. Electrical Events of the Cardiac Cycle.

Different waves can be represented, it depends on where the heart activity is measured. Each version called *lead* can be seen as the representation of the same phenomenon from different angles. Professional ECG sensors have a maximum of 12 leads by applying on the patient 4 electrodes on the limbs and 6 electrodes on the chest. The target of this project is to be able to acquire a heart signal deriving from the first lead, that is, the one received from 2 electrodes positioned, one in the right hand and the other in the left hand. The result of this monitoring is a wave that holds within it, a quantity of information useful for a first diagnosis of the functioning of the heart.

1.1 Einthoven triangle and leads description

Willem Einthoven was the inventor of the electrocardiogram. Its idea, behind this discovery, can be summarized in some points:

- the thorax is a homogeneous spherical conductor with the heart in the center;
- cardiac electrical forces are generated in the center of the conductor and the resulting of these forces can be represented by a unique vector;
- the limb-joint points are the vertices of an inscribed equilateral triangle in the longitudinal section of the spherical thorax, *Einthoven Triangle*, because equidistant and lying on the same plane;
- at any moment, the potential differences recorded by pairs of electrodes placed at the top of the triangle, represent the projections of the resulting heart vector, on the lines joining the electrodes, *leads*;
- the amplitude of the P, QRS and T waves, measured on the track recorded in each derivation corresponds to the projection of the vector it represents respectively: atrial, ventricular and repolarization activation ventricular.

1.1.1 Einthoven triangle

Considering the Einthoven Triangle these derivation has been obtained:

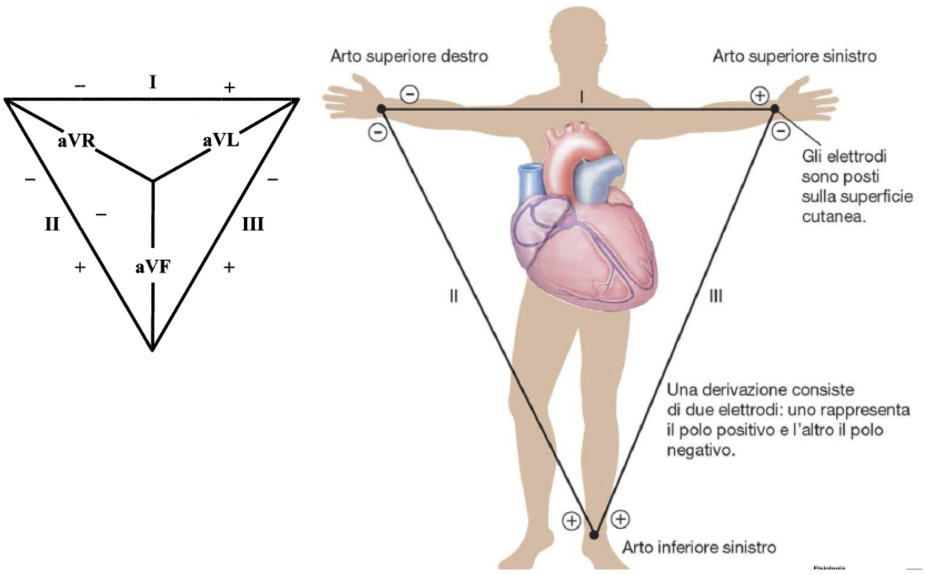


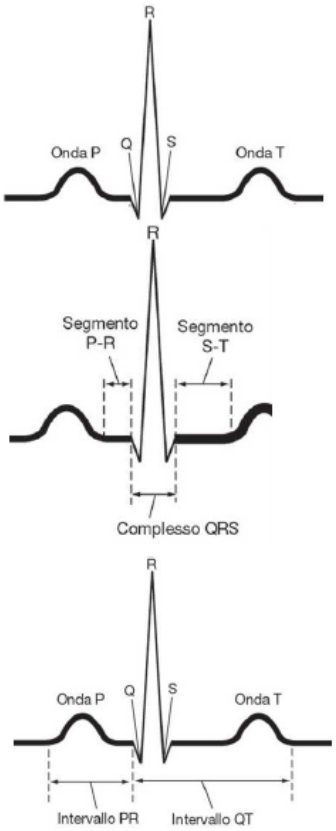
Figure 1.3. Einthoven Triangle.

$$LeadI = V_I = La - Ra \tag{1.1}$$
$$LeadII = V_{II} = LL - Ra \tag{1.2}$$
$$LeadIII = V_{III} = LL - La \tag{1.3}$$
$$aVR = Ra - (La + LL)/2 \tag{1.4}$$
$$aVL = La - (Ra + LL)/2 \tag{1.5}$$
$$aVF = LL - (Ra + La)/2 \tag{1.6}$$

This project aims to develop a Lead I extraction by detecting the tensions on the wrist and finger of the opposite hand.

1.1.2 ECG Graphical Waves Representation

In this subsection the meaning of each individual stretch of wave is briefly described.



Event	Duration [sec]	Amplitude [mV]	Description
P wave	0.07-0.12	0.2-0.4	Atrial depolarization
QRS ensamble	0.06-0.10	1.0-2.0	Ventricular depolarization
T wave	0.18-0.20	0.4-0.5	Ventricular repolarization
P-R lapse	0.12-0.20		Atrioventricular conduction time
Q-T lapse	0.40-0.42		Time ventricular depolarization and repolarization
S-T lapse	0.30-0.34		Time from the end of ventricular depolarization to the beginning of ventricular repolarization
R-R lapse	0.8-0.9		Cardiac cycle duration

Figure 1.4. Graphical Waves Representation.

1.2 Pulse Oximetry and Heart Rate

Pulse oximetry is a noninvasive method of measuring an individual's blood oxygen saturation levels. Oxygen saturation levels, referring to the relationship between oxygenated hemoglobin and total hemoglobin in the blood, can help detect hypoxemia, deterioration of organ function and even cardiac arrest. The Pulse Oximetry measurement can be done in two way [19]:

- **Transmissive Pulse Oximetry:** LEDs transmit light of specific wavelengths through the tissue, which is absorbed by photodetectors on the other end;
- **Reflective Pulse Oximetry:** LEDs transmit light of specific wavelengths through the tissue, which is absorbed by photodetectors on the same side; the reflected signal is monitored for changes in light absorption (PPG photoplethysmography).

The MAM86161 sensor performe a Reflective Pulse Oximetry. Typical signal collected by this component is shown below.

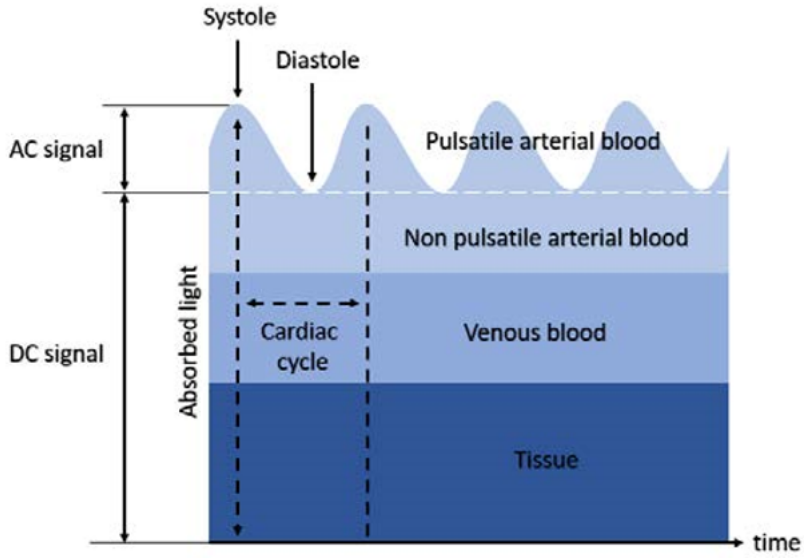


Figure 1.5. DC and AC Component of a PPG Signal. [21]

1.2.1 SpO_2 Measurement

In order to measure the SpO_2 two different wavelegth LEDs must be employ: RED LED and IR LED. Since the DC and AC components of the two LEDs have different amplitudes, the ratio R can be calculated [19]:

$$R = \frac{\frac{AC_{Red}}{DC_{Red}}}{\frac{AC_{IR}}{DC_{IR}}} \tag{1.7}$$

After calculating the value of R, can be easily calculate the value of SpO_2 with a linear approximation derived from a best-fit straight-line approximation of SpO2 vs. R data [20] between the R-range of 0.4 to 3.4:

$$SpO_2 = 104 - 17 \cdot R \tag{1.8}$$

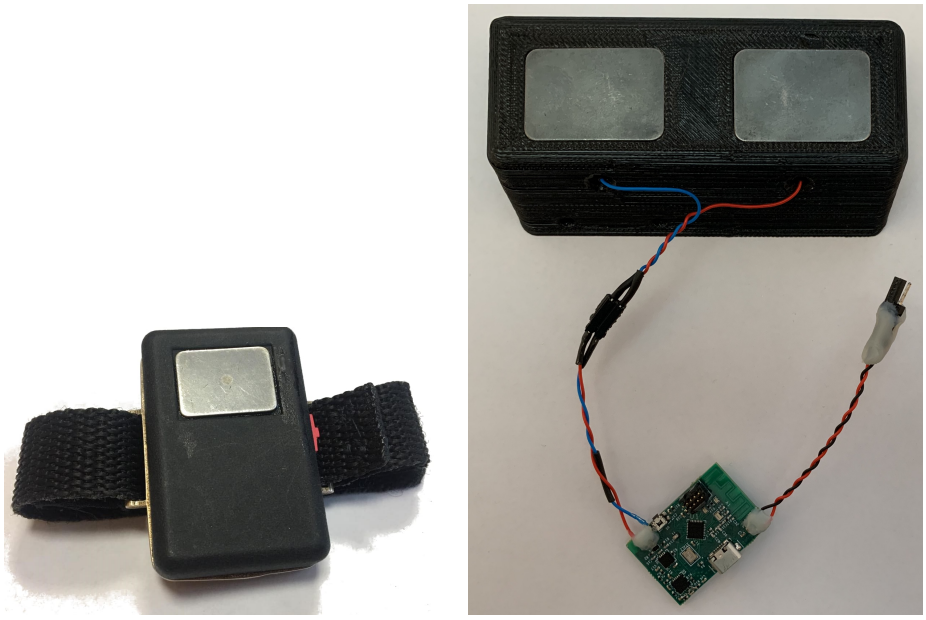
1.3 State of Art

Nowadays it has become very important to monitor some vital parameters through the use of wearable devices and to save these data on external devices such as mobile phones or tablets. There are many devices that can perform the same measurement of the device show in this Thesis. The most known one can be the device produced by Apple so the *Apple Watch* 1.3.

Figure 1.6. Apple Watch Series 5.

This, however, in addition to having the same medical functions as the device presented in this thesis, has many more features that resemble a mobile phone; in addition it has a very high cost. The purpose of this thesis is to create an object accessible to everyone, therefore with low costs.

Before this project there are others old version of the device developed by other students. An example is the Vital EKG device, the improvement that have tried to bring concerns the size of the object, we have tried to make it smaller, for doing this it has been starting from the "Smart wearable wrist ECG with BLE interface" project. The last project before mentioned, is used to give an idea of the small dimension, but it must be improved from the side of disturbances caused by $50Hz$. Also the App has been reused and modified.



(a) ECG Watch device

(b) Smart wearable wrist ECG with BLE interface project

Figure 1.7. Old ECG Device Versions.

Part II

Implementation

Chapter 2

Analysis of the Task

2.1 Executive Summary

The aim of this project is the creation of a wearable device that can measure ECG and PPG signal. For doing this it has been started from a previous version of the device able to do the ECG acquisition; unfortunately this acquisition as many problems as reported in Figure 2.1. As can be seen from the graph below, the signal power spectrum taken into consideration has major disturbances caused by the power supplies that surround the device, that are disturbances multiple of $50Hz$.

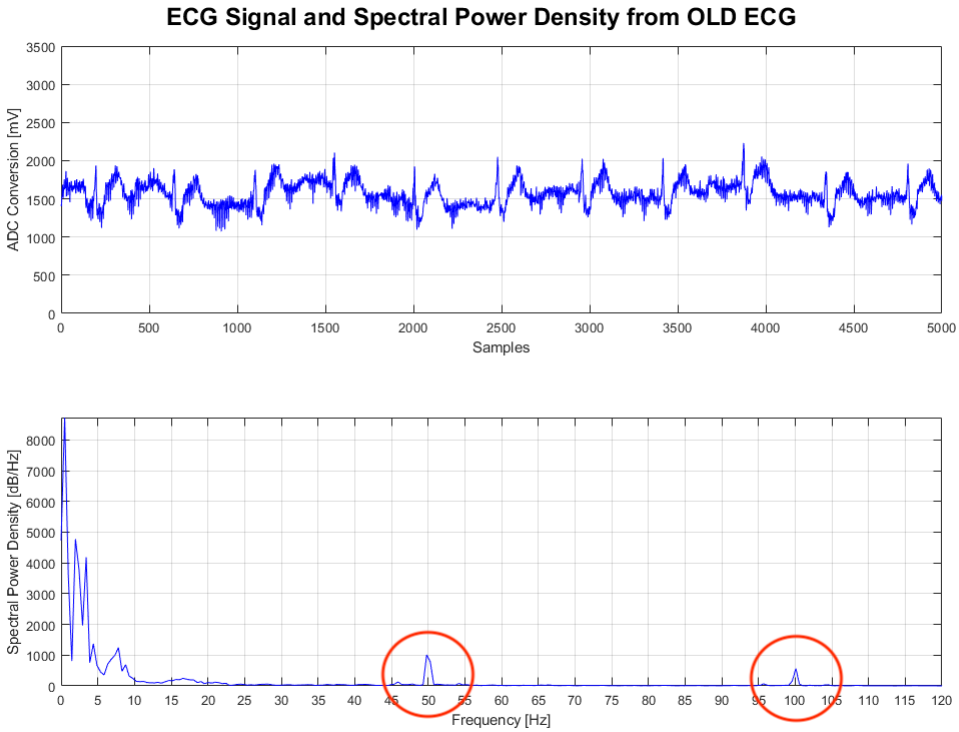


Figure 2.1. Acquisition from OLD ECG Device.

So the first goal of the project was to solve the $50Hz$ problem by introducing a notch filter. This however involved the redesign of the entire front end. In MATLAB chapter, this phase is described more carefully.

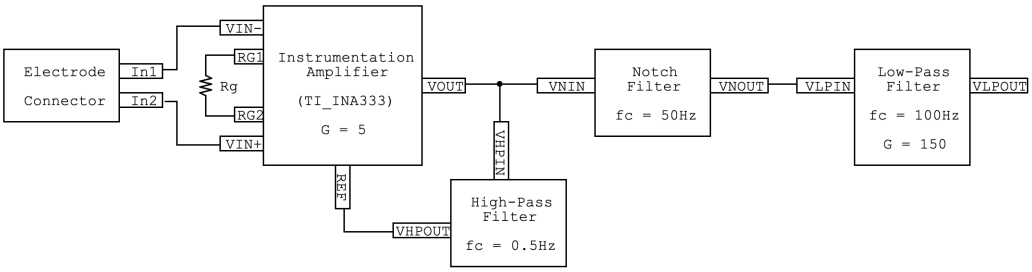


Figure 2.2. Block Diagram of the NEW ECG Front-End.

In Figure 6.1 is represented the simple Block Diagram of the NEW ECG

Front-End designed. Some requirements must be respected to design the Analog Front-End:

- amplification of a factor approximately 1000 (60db) only in the useful band:
  - for frequency monitoring including 0.05 - 50Hz (project case);
  - for frequency diagnostics up to 1kHz;
- high input impedance to prevent the unknown impedance of the electrode from creating a signal partition, attenuating it;
- it must reject a strong network noise (50Hz) of amplitude comparable to the signal;
- safety specifications: according to the law, the current flowing on the patient must be less than 10µA (higher currents increase the incidence of fibrillation).

For doing this it has been used the INA333 Instrumentation Amplifier with this characteristics:

- low cost, low amplification of the differential signals and low power consumption power;
- Amplification selectable through external R_G:

$$G_{Instr.Ampl.} = 1 + \left(\frac{100k\Omega}{R_G}\right) = 5 \tag{2.1}$$

- signal common mode rejection ratio, including mains frequency and its harmonics (CMRR higher than 100dB).

Chapter 3

MATLAB

MATLAB is a useful tool for testing the behaviour of a schematic before mounting the circuit in reality. In this project it has been useful during the design of both hardware and software filters. Starting from a reference circuit, a transfer function can be calculated and through this program can be displayed Bode Diagrams and the Step and Impulse responses, so as to verify the stability of the circuit.

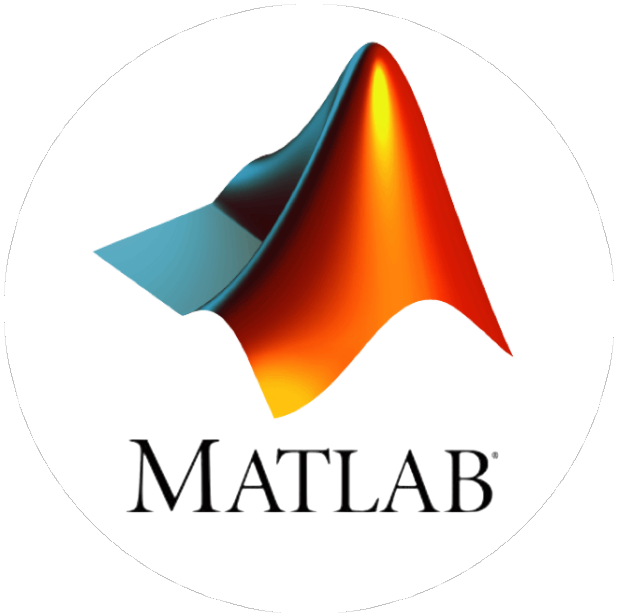


Figure 3.1. MATLAB Logo.

3.1 Hardware Filters

In this section is shown the design of the hardware filters of the ECG analog front-end. Before start the hardware design it has been perform a MATLAB simulation of the filter transfer functions in order to choose the correct sizing of the real electronic components. This test starts from the study of the spectral power density of a typical ECG signal described in the previous chapter.
it has also been wanted to design two different *Version A* and *Version B* filters in order to have comparisons between different Front-Ends. In fact, results relating to these two types of filtering has been reported.

3.1.1 High Pass Filter

The High Pass Filter has been used to remove all the disturbances in a very low frequencies around the continuous, due to little muscle movement during the respiration phase. This filtering has been implemented through use of first order filter (integrator) put in feed-back on the differential amplifier as shown in the Figure 3.2. Starting from the cutting frequency formula of an

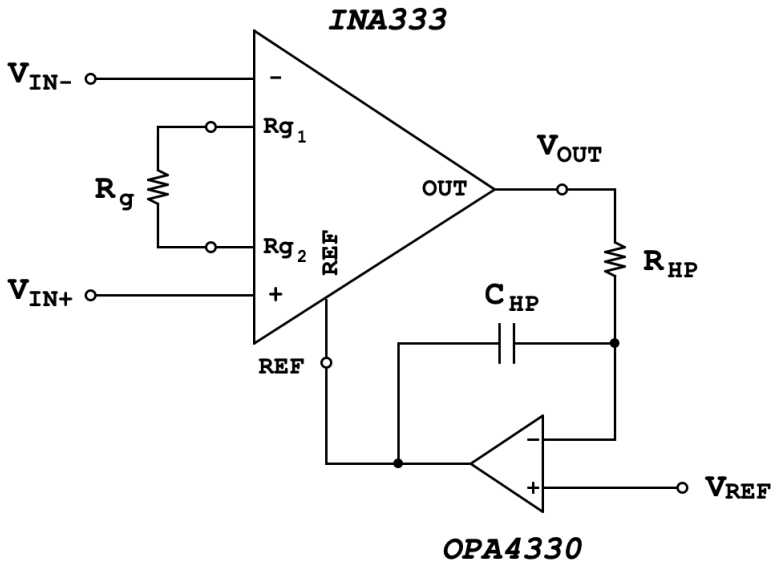


Figure 3.2. High Pass Filter Schematic.

High Pass Filter, resistance and capacitance values have been calculated as shown below:

$$f_{HP} = \frac{1}{2 \cdot \pi \cdot r_{HP} \cdot C_{HP}} \tag{3.1}$$

The cutting frequency wanted is:

$$f_{HP_{ideal}} = 0{,}5Hz \tag{3.2}$$

and the resistance that has been chosen according to E24 series is:

$$R_{HP_{ideal}} = R_{HP_{real}} = 68k\Omega \tag{3.3}$$

So, the calculated capacitance is:

$$C_{HP_{ideal}} = \frac{1}{2 \cdot \pi \cdot f_{HP_{ideal}} \cdot R_{HP_{ideal}}} = \frac{1}{2 \cdot \pi \cdot 0{,}5Hz \cdot 68k\Omega} = 4.68\mu F \quad (3.4)$$

and the chosen value according to E24 series is:

$$C_{HP_{real}} = 4.7\mu F \quad (3.5)$$

So, the real cutting frequency becomes:

$$f_{HP_{real}} = \frac{1}{2 \cdot \pi \cdot R_{HP_{real}} \cdot C_{HP_{real}}} = \frac{1}{2 \cdot \pi \cdot 68k\Omega \cdot 4.7\mu F} = 0.498Hz \quad (3.6)$$

The transfer function of the filter in Figure 3.2 has been calculated as shown below:

$$REF = V_{OUT} \cdot \left(- \frac{1}{s \cdot C_{HP} \cdot R_{HP}} \right) + V_{REF} \quad (3.7)$$

$$\begin{aligned} V_{OUT} &= [V_{IN+} - V_{IN-}] + REF = \\ &= [V_{IN+} - V_{IN-}] + V_{REF} - V_{OUT} \cdot \left(\frac{1}{s \cdot C_{HP} \cdot R_{HP}} \right) \end{aligned} \quad (3.8)$$

$$V_{OUT} \cdot \left(1 + \frac{1}{s \cdot C_{HP} \cdot R_{HP}} \right) = [V_{IN+} - V_{IN-}] + V_{REF} \quad (3.9)$$

$$\begin{aligned} V_{OUT} &= \frac{[V_{IN+} - V_{IN-}] + V_{REF}}{\left(1 + \frac{1}{s \cdot C_{HP} \cdot R_{HP}} \right)} = \\ &= \frac{s \cdot C_{HP} \cdot R_{HP} \cdot \{[V_{IN+} - V_{IN-}] + V_{REF}\}}{(s \cdot C_{HP} \cdot R_{HP} + 1)} \end{aligned} \quad (3.10)$$

$$H_{HPF}(s) = \frac{V_{OUT}}{[V_{IN+} - V_{IN-}] + V_{REF}} = \frac{s \cdot C_{HP} \cdot R_{HP}}{(s \cdot C_{HP} \cdot R_{HP} + 1)} \quad (3.11)$$

At this point, after reporting the formulas in a MATLAB script it has been performed some tests in order to evaluate the filter's quality. It has been presented two result: the ideal one in which the used values derives from the initial values founded from formulas 3.73, 3.3 and 3.4; the second one in which the value of resistance and capacitance are the real component value, according to E24 series, 3.3, 3.5 so that it has been calculated and evaluated the real cutting frequency 3.6.

- **Ideal High Pass Filter Result and MATLAB Code**
  In this subsection the ideal MATLAB results obtained and the code used have been presented.

  MATLAB Results

$f_{HP_{ideal}}$[Hz]	$R_{HP_{ideal}}$[kΩ]	$C_{HP_{ideal}}$[µF]
0,5	68	4.68

Table 3.1. Ideal values used during this simulation.

MATLAB Simulation

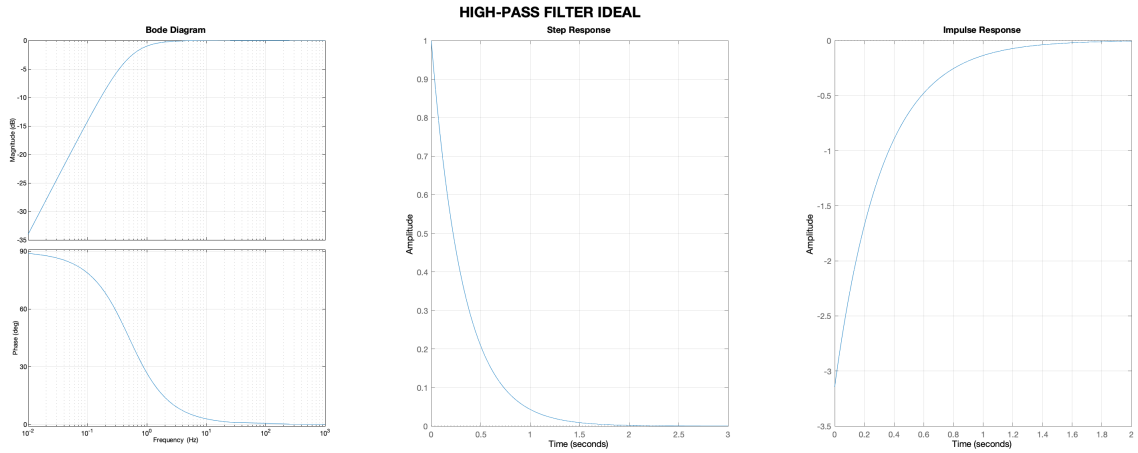


Figure 3.3. High Pass Filter Ideal Response.

MATLAB Code

```
%% High(Low)_Pass_Active_Filter [Analog Filter GUIDE]
f_HP = 0.5; %Hz
R_HP = 68e3; %Ohm

C_HP = 1/(2*pi*f_HP*R_HP)

G_HP = (s*C_HP*R_HP)/(s*C_HP*R_HP+1);
options = bodeoptions;
options.FreqUnits = 'Hz';
options.Xlim = [0.01, 1000];

figure('position', [2,742,2560,614],'NumberTitle', 'off', 'Name', 'HIGH-PASS FILTER
    Ideal')

subplot(1,3,1)
bode(G_HP, options);
title('Bode Diagram', 'fontweight', 'bold', 'fontsize', 11);
grid on;

subplot(1,3,2)
step(G_HP);
grid on;

subplot(1,3,3)
impulse(G_HP);
grid on;

sgtitle('HIGH-PASS FILTER IDEAL', 'fontweight', 'bold', 'fontsize', 18);
```

- **Real High Pass Filter Result and MATLAB Code**
  In this subsection the real MATLAB results obtained and the code used have been presented. This result are the ones that come closest to reality, also because real values of resistors and capacitors have tolerances that make the real results differ from those calculated in these tests.

MATLAB Results

$f_{HP_{ideal}}$[Hz]	$R_{HP_{ideal}}$[kΩ]	$C_{HP_{ideal}}$[µF]
0.498	68	4.7

Table 3.2. Ideal values used during this simulation.

MATLAB Simulation

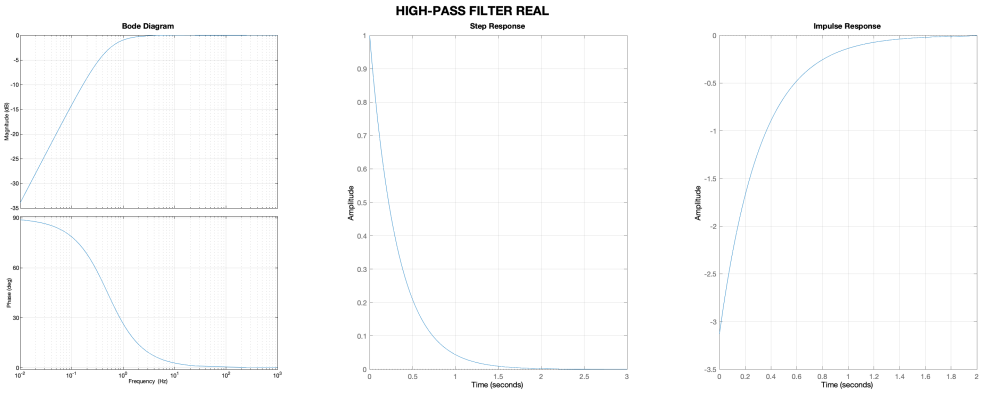


Figure 3.4. High Pass Filter Real Response.

MATLAB Code

```
%% High(Low)_Pass_Active_Filter [Analog Filter GUIDE]
R_HP_real = 68e3; %Ohm
C_HP_real = 4.7e-6; %F C_HP = 4.6810e-06
f_HP_real = 1/(2*pi*R_HP_real*C_HP_real)

G_HP_real = (s*C_HP_real*R_HP_real)/(s*C_HP_real*R_HP_real+1);
options = bodeoptions;
options.FreqUnits = 'Hz';
options.Xlim = [0.01, 1000];

figure('position', [2,742,2560,614],'NumberTitle', 'off', 'Name', 'HIGH-PASS FILTER
    Real Component')

subplot(1,3,1)
bode(G_HP_real, options);
title('Bode Diagram', 'fontweight', 'bold', 'fontsize', 11);
grid on;

subplot(1,3,2)
step(G_HP_real);
grid on;

subplot(1,3,3)
impulse(G_HP_real);
grid on;

sgtitle('HIGH-PASS FILTER REAL', 'fontweight', 'bold', 'fontsize', 18);
```

3.1.2 Twin-T Notch Filter

The Notch Filter also called Band Stop Filter is able to reject a set of frequency close to the cutting frequency and in this project has been used to remove all the disturbances due to equipment around the device and so the frequency under investigation is the power frequency $50Hz$. In the Band Stop Filter passes all frequency from $0Hz$ to lower cut-off frequency and from the higher cut-off frequency up as shown in the Figure 3.5. This filter

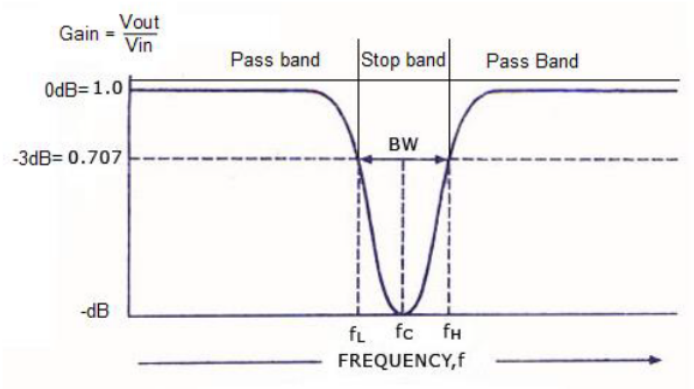


Figure 3.5. Notch Filter Response.

is obtained connecting Low Pass Filter in parallel to an High Pass Filter in order to not have overlapping in the produced frequency response Figure 3.6. [1]

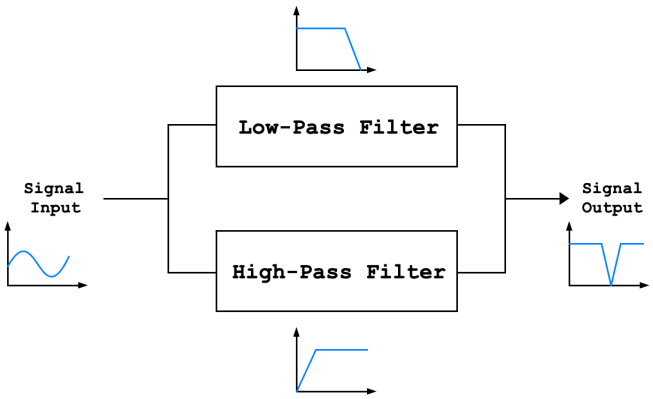


Figure 3.6. Notch Filter Block Diagram.

The Twin-T Filter can be implemented in two ways:

- **with NO feedback (Version A)**: in this case the Q factor (notch depth) is fixed to 0.25;
- **with feedback (Version B)**: in this case the ratio set by:

$$R_{N1}/R_{N2}$$

determine the Q factor value; the maximum depth is reached when the resistor, the capacitor and the Op-Amp in feedback are replaced by a short circuit. [2]

This two type of filter has been studied and designed in order to decide which was the best for this type of application.

Notch Filter Version A

Twin-T Filter with NO feedback shown in Figure 3.9 has been composed by a passive Twin-T followed by an Op-Amp in voltage follower configuration.

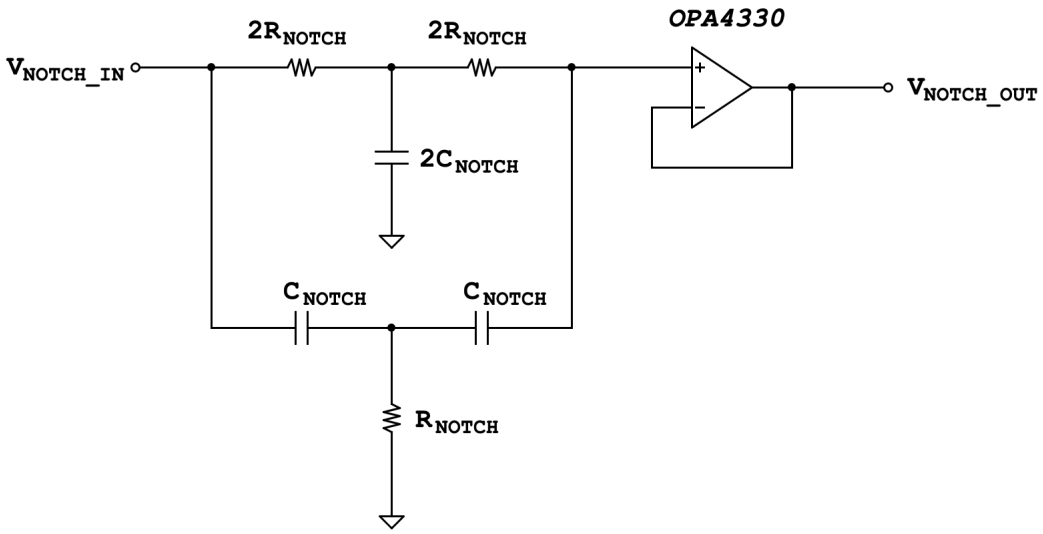


Figure 3.7. Notch Filter Version A Schematic

Parameters of this filter have been calculated using a system of equations reported below: [3]

$$\begin{cases} f_{NOTCH} = \dfrac{1}{2 \cdot \pi} \cdot \sqrt{\dfrac{\dfrac{1}{C_{NOTCH}} + \dfrac{1}{C_{NOTCH}}}{2C_{NOTCH} \cdot 2R_{NOTCH} \cdot 2R_{NOTCH}}} \\ f_{NOTCH} = \dfrac{1}{2 \cdot \pi} \cdot \sqrt{\dfrac{1}{C_{NOTCH} \cdot C_{NOTCH} \cdot R_{NOTCH} \cdot (4R_{NOTCH})}} \end{cases} \tag{3.12}$$

The cutting frequency wanted is:

$$f_{NOTCH_{ideal}} = 50Hz \tag{3.13}$$

and the capacitances that has been chosen according to E24 series are:

$$C_{NOTCH} = 47nF \tag{3.14}$$
$$2C_{NOTCH} = 100nF \tag{3.15}$$

So, the calculated resistances are:

$$R_{NOTCH} = 36k\Omega \tag{3.16}$$
$$2R_{NOTCH} = 68k\Omega \tag{3.17}$$

So, the real cutting frequency becomes:

$$f_{NOTCH_{real}} = 48.40Hz \tag{3.18}$$

The transfer function of this Twin-T Notch Filter is shown below: [4]

$$\frac{vo}{vi} = \frac{s^3 + s^2\frac{1}{C1}\left(\frac{1}{R1}+\frac{1}{R2}\right) + s\frac{1}{C1R1R2}\left(\frac{1}{C3}+\frac{1}{C2}\right) + \frac{1}{C1C2C3R1R2R3}}{s^3 + s^2\left(\frac{1}{C1R1}+\frac{1}{C1R2}+\frac{1}{C2R2}+\frac{1}{C2R3}+\frac{1}{C3R2}\right) + s\left(\frac{1}{C2R3C1R1}+\frac{1}{C2R3C1R2}+\frac{1}{C3C1R1R2}+\frac{1}{C2C1R1R2}+\frac{1}{C2C3R3R2}\right) + \frac{1}{C1C2C3R1R2R3}}$$

Figure 3.8. Twin-T Filter Transfer Function.

Where,

$$R1 = 2R_{NOTCH} \tag{3.19}$$
$$R2 = 2R_{NOTCH} \tag{3.20}$$
$$R3 = R_{NOTCH} \tag{3.21}$$
$$C1 = 2C_{NOTCH} \tag{3.22}$$
$$C2 = C_{NOTCH} \tag{3.23}$$
$$C3 = C_{NOTCH} \tag{3.24}$$

At this point, after reporting the formulas in a MATLAB script it has been performed some tests in order to evaluate the filter's quality. Differently from the High Pass Filter, before presented, in this case has been presented only the result (Figure 3.9) in which are used the real component value, according to E24 series, 3.16 and 3.14. As can be seen from the figure above, the

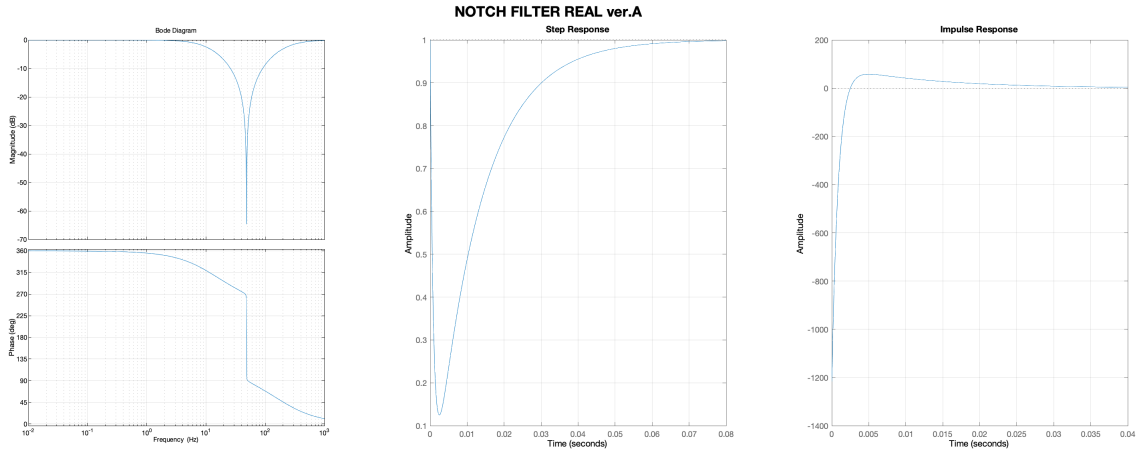


Figure 3.9. Notch Filter Version A Schematic.

behavior of the phase of the filter has a somewhat strange trend. This was one of some reasons why this circuit was abandoned by the project.

Notch Filter Version B

Twin-T Filter with feedback shown in Figure 3.10 has been composed by a passive Twin-T followed by an Op-Amp in voltage follower configuration; the output is than connected with positive feedback to the reference node between R_{NOTCH} and $2C_{NOTCH}$.

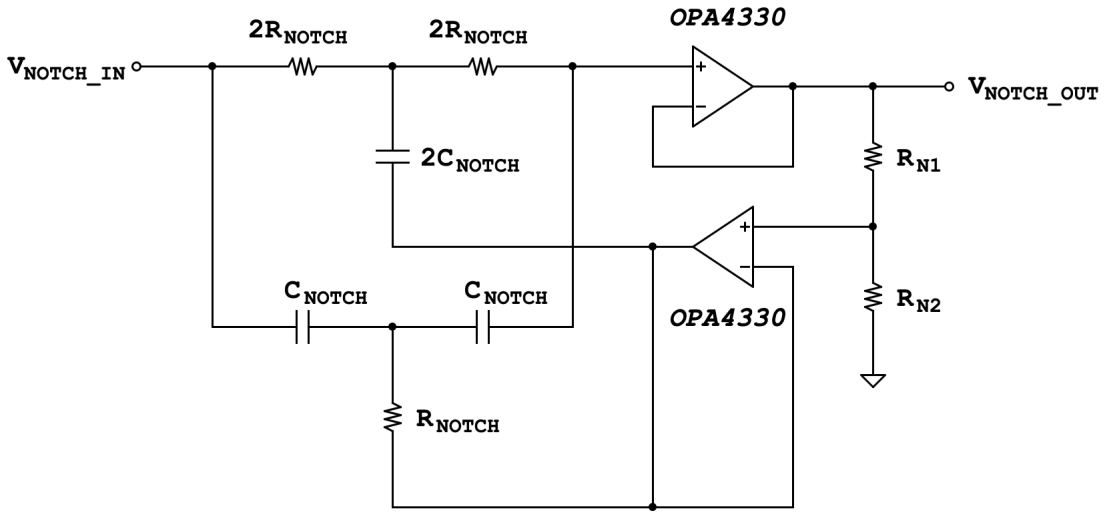


Figure 3.10. Notch Filter Version B Schematic.

For this version of the Twin-T Filter has been applied another formulas in order to find real component value and filter transfer function. Starting from the cutting frequency, resistance and capacitance values have been calculated as shown below:

$$f_{NOTCH} = \frac{1}{2 \cdot \pi \cdot 2R_{NOTCH} \cdot C_{NOTCH}} \tag{3.25}$$

The cutting frequency wanted is:

$$f_{NOTCH_{ideal}} = 50Hz \tag{3.26}$$

and the capacitance that has been chosen according to E24 series is:

$$C_{NOTCH_{ideal}} = C_{NOTCH_{real}} = 150nF \tag{3.27}$$

$$2C_{NOTCH_{ideal}} = 2C_{NOTCH_{real}} = 300nF \tag{3.28}$$

So, the calculated resistance is:

$$\begin{aligned} 2R_{NOTCH_{ideal}} &= \frac{1}{2 \cdot \pi \cdot f_{NOTCH_{ideal}} \cdot C_{NOTCH_{ideal}}} = \\ &= \frac{1}{2 \cdot \pi \cdot 50Hz \cdot 150nF} = 21.22k\Omega \end{aligned} \tag{3.29}$$

and the chosen values according to E24 series is:

$$2R_{NOTCH_{real}} = 21.3k\Omega \tag{3.30}$$

$$R_{NOTCH_{real}} = 10.7k\Omega \tag{3.31}$$

So, the real cutting frequency becomes:

$$\begin{aligned} f_{NOTCH_{real}} &= \frac{1}{2 \cdot \pi \cdot 2R_{NOTCH} \cdot C_{NOTCH}} \\ &= \frac{1}{2 \cdot \pi \cdot 21.3k\Omega \cdot 150nF} = 49.81Hz \end{aligned} \tag{3.32}$$

In this type of circuit is possible to evaluate the value of R_{N1} and R_{N2} (feedback ratio), K (feedback factor) and Q (quality factor) that, in turn, determine the notch depth all starting from wanted BW value (Bandwidth). So, the values chosen for this project are:

$$BW_{NOTCH} = 20Hz \tag{3.33}$$

$$R_{N2_{ideal}} = R_{N2_{real}} = 10k\Omega \tag{3.34}$$

Than:

$$Q_{NOTCH_{ideal}} = \frac{f_{NOTCH_{ideal}}}{BW_{NOTCH}} = 10 \tag{3.35}$$

$$K_{NOTCH_{ideal}} = 1 - \left(\frac{1}{4 \cdot Q_{NOTCH_{ideal}}} \right) = 0.98 \tag{3.36}$$

$$R_{N1_{ideal}} = (1 - K_{NOTCH_{ideal}}) \cdot \left(\frac{R_{N2}}{K_{NOTCH_{ideal}}} \right) = 1.11k\Omega \tag{3.37}$$

So, the real values become:

$$R_{N1_{real}} = 1.1k\Omega \tag{3.38}$$

$$Q_{NOTCH_{real}} = \frac{f_{NOTCH_{real}}}{BW_{NOTCH}} = 9.96 \tag{3.39}$$

$$K_{NOTCH_{ideal}} = 1 - \left(\frac{1}{4 \cdot Q_{NOTCH_{real}}} \right) = 0.97 \tag{3.40}$$

The transfer function of the filter in Figure 3.10 is shown below: [2]

$$H_N(s) = \frac{s^2 + \left(\frac{1}{2R_{NOTCH} \cdot C_{NOTCH}}\right)^2}{s^2 + 4s \cdot \frac{2R_{NOTCH}}{C_{NOTCH}} \cdot \left(1 - \frac{R_{N2}}{R_{N1}+R_{N2}}\right) + \left(\frac{1}{2R_{NOTCH} \cdot C_{NOTCH}}\right)^2} \quad (3.41)$$

At this point, after reporting the formulas in a MATLAB script it has been performed some tests in order to evaluate the filter's quality. It has been presented two result: the ideal one in which the used values derives from the initial values founded from formulas 3.26, 3.29 and 3.27; the second one in which the value of resistance and capacitance are the real component value, according to E24 series, 3.30, 3.27 so that it has been calculated and evaluated the real cutting frequency 3.32.

- **Ideal Twin-T Filter Vers.B Result and MATLAB Code**
  In this subsection the ideal MATLAB results obtained and the code used have been presented.

  MATLAB Results

$f_{N_{ideal}}$[Hz]	$R_{N_{ideal}}$[kΩ]	$2R_{N_{ideal}}$[kΩ]	$C_{N_{ideal}}$[nF]	$C_{N_{ideal}}$[nF]
50	10.61	21.22	150	300

Table 3.3. Ideal values used during this simulation.

MATLAB Simulation

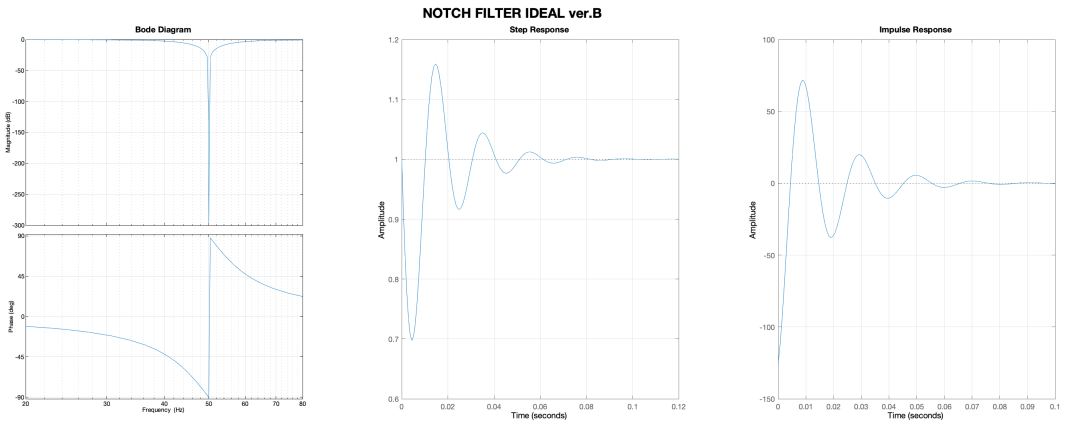


Figure 3.11. Notch Filter Ideal vers.B Response.

MATLAB Code

```
%% Twin_T_Notch_Active_Filter [Analog Filter GUIDE]
f_notch = 50; %Hz
C_notch = 150e-9; %F
R9 = 10e3; %Ohm
BW_notch = 20; %Hz

R_notch = 1/(2*pi*f_notch*C_notch)
Q_notch = f_notch/BW_notch
K_notch = 1-(1/(4*Q_notch))
R8 = (1 - K_notch)*(R9/K_notch)

s = tf('s');
G_notch = ((s^2)+(1/(R_notch*C_notch))^2)/((s^2)+((4*s/(R_notch*C_notch))*(1-(R9/(R8+R9
    ))))+(1/(R_notch*C_notch))^2);

options = bodeoptions;
options.FreqUnits = 'Hz';
options.Xlim = [20,80];

figure('position', [2,742,2560,614],'NumberTitle', 'off', 'Name', 'NOTCH FILTER Ideal')

subplot(1,3,1)
bode(G_notch, options);
title('Bode Diagram', 'fontweight', 'bold', 'fontsize', 11);
grid on;

subplot(1,3,2)
step(G_notch);
grid on;

subplot(1,3,3)
impulse(G_notch);
grid on;

sgtitle('NOTCH FILTER IDEAL ver.B', 'fontweight', 'bold', 'fontsize', 18);
```

- **Real Twin-T Filter Vers.B Result and MATLAB Code**
  In this subsection the real MATLAB results obtained and the code used have been presented. This result are the ones that come closest to reality, also because real values of resistors and capacitors have tolerances that make the real results differ from those calculated in these tests.

MATLAB Results

$f_{N_{real}}$[Hz]	$R_{N_{real}}$[kΩ]	$2R_{N_{real}}$[kΩ]	$C_{N_{real}}$[nF]	$C_{N_{real}}$[nF]
49.81	10.7	21.3	150	300

Table 3.4. Real values used during this simulation.

MATLAB Simulation

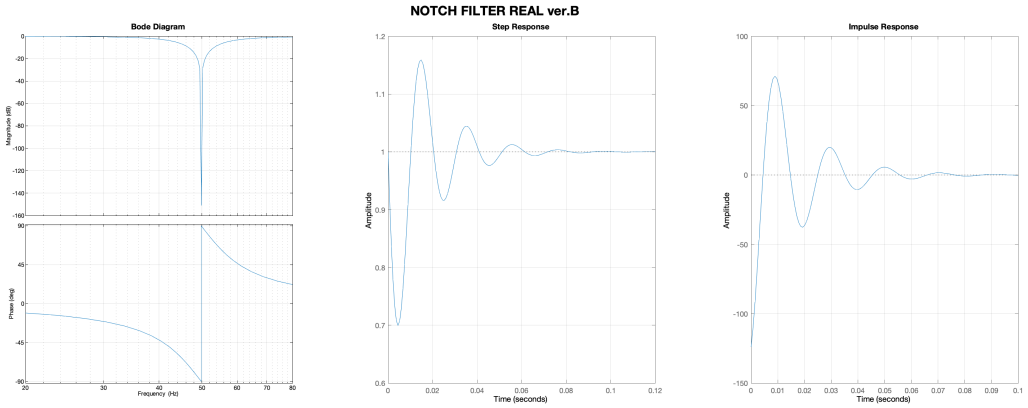


Figure 3.12. Notch Filter Real vers.B Response.

MATLAB Code

```
%% Twin_T_Notch_Active_Filter [Analog Filter GUIDE]
C_notch_real = 150e-9; %F
R9_real = 10e3; %Ohm
R_notch_real = 21.3e3; %Ohm R_notch = 2.1221e+04
R8_real = 1.1e3; %Ohm R8 = 1.1111e+03 esiste anche 1.11kOhm oltre a 1.1kOhm
f_notch_real = 1/(2*pi*R_notch_real*C_notch_real)
Q_notch_real = f_notch_real/BW_notch
K_notch_real = 1-(1/(4*Q_notch_real))

s = tf('s');
G_notch=((s^2)+(1/(R_notch_real*C_notch_real))^2)/((s^2)+((4*s/(R_notch_real*
    C_notch_real))*(1-(R9_real/(R8_real+R9_real))))+(1/(R_notch_real*C_notch_real))^2)

options = bodeoptions;
options.FreqUnits = 'Hz';
options.Xlim = [20,80];
figure('position', [2,742,2560,614],'NumberTitle', 'off', 'Name', 'NOTCH FILTER Real
    Component')

subplot(1,3,1)
bode(G_notch, options);
title('Bode Diagram', 'fontweight', 'bold', 'fontsize', 11);
grid on;

subplot(1,3,2)
step(G_notch);
grid on;

subplot(1,3,3)
impulse(G_notch);
grid on;
sgtitle('NOTCH FILTER REAL ver.B', 'fontweight', 'bold', 'fontsize', 18);
```

3.1.3 Low Pass Filter

The design of this filter depend on the realization of the Twin-T Filter, this fact because the number of Op-Amp that can be used is limited according to the number contained in the component chosen OPA4330. It has inside only four Op-Amp. If it has been used the Twin-T Filter version A, the number of amplifiers available would have been two otherwise only one. In fact it has been reported two version of this Low Pass Filter:

- **Version A:** a circuit with a K gain of 1, this solution needs a final gain in the last block implemented with a non-inverting amplifier;
- **Version B:** a circuit that contain inside the management of the K gain. Moreover, it can be possible to choose different types of transfer functions (behaviours of the filter) and architectures:
  - **Transfer function (behaviour of the filter):**
    - *Butterworth Filter*: filters are optimized for maximally flat magnitude response, gain flatness in bass-band, the attenuation is -3dB at the cutting frequency and above this frequency the attenuation is $20dB/dec$. The negative point is the transient response to pulse input, it can generate overshoot and ringing.
    - *Chebyshev Filter*: filters are optimized to have ripple in pass-band and so that the cut-off frequency is defined as the frequency in which the response falls below this ripple band. Their response to a pulse input is worst than Butterworth filters.
    - *Bessel Filter*: filters are designed to have a maximally flat time delay so they have linear phase response and a good transient response to pulse input, the attenuation is -3dB at the cutting frequency.
  - **Architecture:** second-order low-pass filters.
    - *Sallen-Key Circuit*
    - *Multiple-Feedback Circuit*

It has been chosen the Bessel Filter behaviour and the Sallen-Key circuit and in the following pages has been analysed two versions. [5] Before doing that, other parameters must be introduced:

- **Q (Quality factor)**: is a measure of how the band-pass filter is selective or not, towards a given spread of frequencies.

- **FSF (Frequency Scaling Factor)**: how the cut-off frequency is scaled.

For a Sallen-Key filter shown in the Figure 3.14 and Figure 3.18 they are equal to:

$$Q = \frac{\sqrt{mn}}{m + 1 + mn \cdot (1 - K)} \tag{3.42}$$

$$FSF \cdot f_C = \frac{1}{2 \cdot \pi \cdot R \cdot C \cdot \sqrt{mn}} \tag{3.43}$$

where:

$$R_{LP1} = mR, R_{LP2} = R \tag{3.44}$$

$$C_{LP1} = nC, C_{LP2} = C \tag{3.45}$$

$$K = \frac{R_{LP4} + R_{LP3}}{R_{LP4}} \tag{3.46}$$

The value of Q and FSF are found in the table below: [5]

FILTER ORDER	Stage 1		Stage 2		Stage 3		Stage 4		Stage 5	
	FSF	Q	FSF	Q	FSF	Q	FSF	Q	FSF	Q
2	1.2736	0.5773								
3	1.4524	0.6910	1.3270							
4	1.4192	0.5219	1.5912	0.8055						
5	1.5611	0.5635	1.7607	0.9165	1.5069					
6	1.6060	0.5103	1.6913	0.6112	1.9071	1.0234				
7	1.7174	0.5324	1.8235	0.6608	2.0507	1.1262	1.6853			
8	1.7837	0.5060	2.1953	1.2258	1.9591	0.7109	1.8376	0.5596		
9	1.8794	0.5197	1.9488	0.5894	2.0815	0.7606	2.3235	1.3220	1.8575	
10	1.9490	0.5040	1.9870	0.5380	2.0680	0.6200	2.2110	0.8100	2.4850	1.4150

Figure 3.13. Bessel Filter Table.

The values for both version A and B are:

$$Q = 0.5773 \tag{3.47}$$

$$FSF = 1.2736 \tag{3.48}$$

Low Pass Filter Version A

This type of circuit reported in Figure 3.14 have a K gain of 1, this solution needs a final gain in the last front-end block implemented with a non-inverting amplifier shown in Figure 3.15.

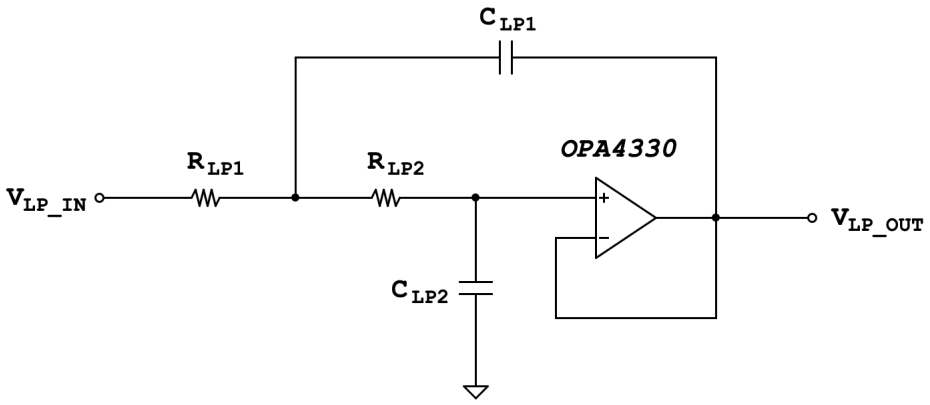


Figure 3.14. Low Pass Filter Version A Schematic.

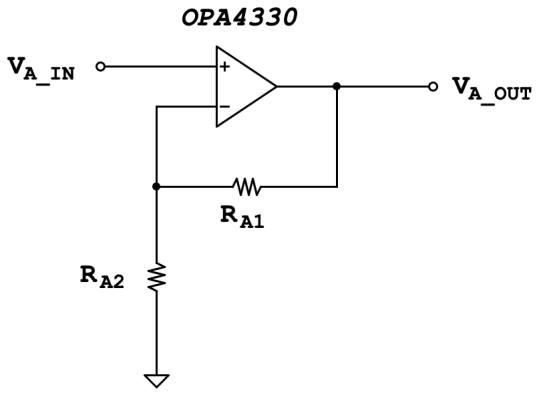


Figure 3.15. Non-Inverting Amplifier Schematic.

The stage reported in the Figure 3.15 is used only to amplify the signal before filtered. The total amplification wanted is:

$$G_{Final} = 650 \tag{3.49}$$

The amplification due to the instrumentation amplifier is:

$$G_{Instr.Ampl.} = 1 + \left(\frac{100k\Omega}{R_{G_{Real}}}\right) = 5.02 \tag{3.50}$$

where:

$$R_{G_{Real}} = 24.9k\Omega \tag{3.51}$$

Choosing R_{A2} it is possible to calculate the value of R_{A1}:

$$R_{A2} = 1k\Omega \tag{3.52}$$

$$R_{A1} = \left(\left(\frac{G_{Final}}{G_{Instr.Ampl.}}\right) \cdot R_{A2}\right) - R_{A2} = 128.6k\Omega \tag{3.53}$$

So,

$$R_{A1} = 130k\Omega \tag{3.54}$$

$$G_{RealFinal} = 657.1 \tag{3.55}$$

This last value is not the real one because of the tolerances of the components and below are reported the MATLAB code used to perform these calculus:

```
%% Operational Amplifier Last Stage
R13_real = 1e3; %Ohm
R3_real = 24.9e3; %Ohm
G_final = 650; %1
G_instr_ampl = 1+(100e3/R3_real)
R12 = ((G_final/G_instr_ampl)*R13_real) - R13_real
R12_real = 130e3; %Ohm
G_final_real = (1+(R12_real/R13_real))*G_instr_ampl
```

Below are reported all the mathematical passages used to decide the values of the components that has been used in order to have a cut-off frequency wanted of:

$$f_{LP_{ideal}} = \frac{100Hz}{FSF} \tag{3.56}$$

It has been applied a system of equations to find the values of m and n:

$$\begin{cases} FSF \cdot f_{LP_{ideal}} &= \dfrac{1}{2 \cdot \pi \cdot R \cdot C \cdot \sqrt{mn}} \\ Q &= \dfrac{\sqrt{mn}}{m+1} \end{cases} \tag{3.57}$$

It has been decided:

$$R = 10k\Omega \tag{3.58}$$
$$C = 100nF \tag{3.59}$$

What it has been obtained is:

$$t = (FSF \cdot f_{LP_{ideal}} \cdot 2 \cdot \pi \cdot R \cdot C) \tag{3.60}$$
$$n = \frac{Q}{(t - Q \cdot t^2)} = 1.44 \tag{3.61}$$
$$m = \frac{1}{t^2 \cdot n} = 1.76 \tag{3.62}$$

So,

$$R_{LP1_{ideal}} = 17.56k\Omega \tag{3.63}$$
$$R_{LP2_{ideal}} = 10k\Omega \tag{3.64}$$
$$C_{LP1_{ideal}} = 144nF \tag{3.65}$$
$$C_{LP2_{ideal}} = 100nF \tag{3.66}$$

The real values become:

$$R_{LP1_{real}} = 18k\Omega \tag{3.67}$$
$$R_{LP2_{real}} = 10k\Omega \tag{3.68}$$
$$C_{LP1_{real}} = 150nF \tag{3.69}$$
$$C_{LP2_{real}} = 100nF \tag{3.70}$$
$$f_{LP_{real}} = 97.95Hz \tag{3.71}$$

The transfer function of the filter in Figure 3.14 is shown below:

$$H_{LP}(s) = \frac{1}{s^2 \cdot (mR^2 \cdot nC^2) + s \cdot ((mR \cdot C) + (R \cdot nC)) + 1} \tag{3.72}$$

At this point, after reporting the formulas in a MATLAB script it has been-performed some tests in order to evaluate the filter's quality as in previous cases.

- **Ideal Low-Pass Filter Vers.A Result and MATLAB Code**
  In this subsection the ideal MATLAB results obtained and the code used have been presented.

  MATLAB Results

$f_{LP_{ideal}}$[Hz]	$R_{LP1_{ideal}}$[kΩ]	$R_{LP2_{ideal}}$[kΩ]	$C_{LP1_{ideal}}$[nF]	$C_{LP2_{ideal}}$[nF]
100	17.56	10	144	100

Table 3.5. Real values used during this simulation.

MATLAB Simulation

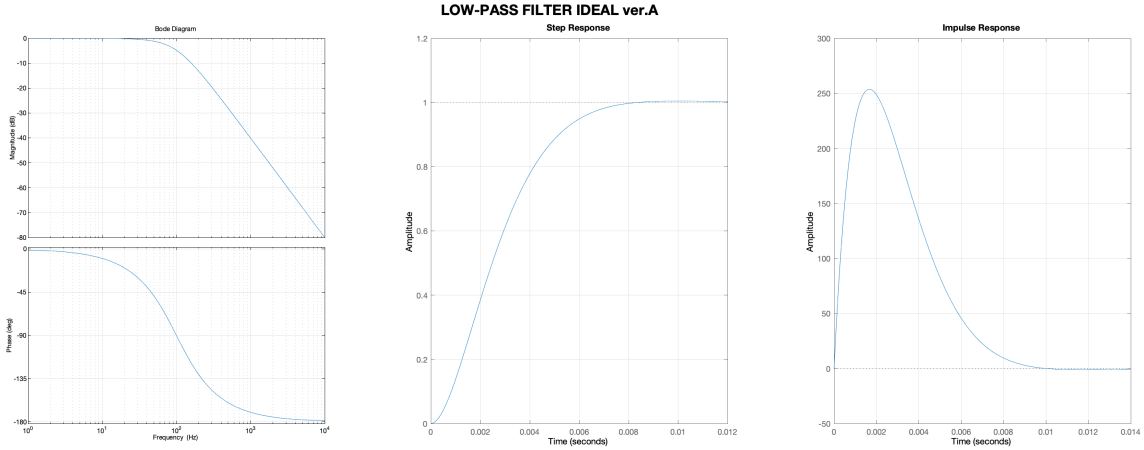


Figure 3.16. Low-Pass Filter Ideal vers.A Response.

MATLAB Code

```
%% Sallen_Key_Low_Pass_Filter (Second_Order_Bessel_Filter) [TI GUIDE]
FSF_LP = 1.2736; %1
Q_LP = 0.5773; %1
f_LP = 100/FSF_LP; %Hz
C_LP = 100e-9; %F
R_LP = 10e3; %Ohm
K_LP = 1; %1

t = (FSF_LP*f_LP*2*pi*R_LP*C_LP);

n = Q_LP/(t-(Q_LP*(t^2)))
m = 1/((t^2)*n)

R5 = m*R_LP
R6 = R_LP
C9 = n*C_LP
C10 = C_LP
f_LP_idea1 = 1/(FSF_LP*2*pi*C_LP*R_LP*sqrt(m*n))

s = tf('s');
G_LP = (1)/((s^2)*(R5*R6*C9*C10)+s*((R5*C10)+(R6*C10))+1);
options = bodeoptions;
options.FreqUnits = 'Hz';
options.Xlim = [1,10000];

```

```
26 figure('position', [2,742,2560,614])
27
28 subplot(1,3,1)
29 bode(G_LP, options);
30 grid on;
31
32 subplot(1,3,2)
33 step(G_LP);
34 grid on;
35
36 subplot(1,3,3)
37 impulse(G_LP);
38 grid on;
39
40 sgtitle('LOW-PASS FILTER IDEAL ver.A', 'fontweight', 'bold', 'fontsize', 18);
```

- **Real Low-Pass Filter Vers.A Result and MATLAB Code**
  In this subsection the real MATLAB results obtained and the code used have been presented. This result are the ones that come closest to reality, also because real values of resistors and capacitors have tolerances that make the real results differ from those calculated in these tests.

MATLAB Results

$f_{LP_{real}}$[Hz]	$R_{LP1_{real}}$[kΩ]	$R_{LP2_{real}}$[kΩ]	$C_{LP1_{real}}$[nF]	$C_{LP2_{real}}$[nF]
97.95	18	10	150	100

Table 3.6. Real values used during this simulation.

MATLAB Simulation

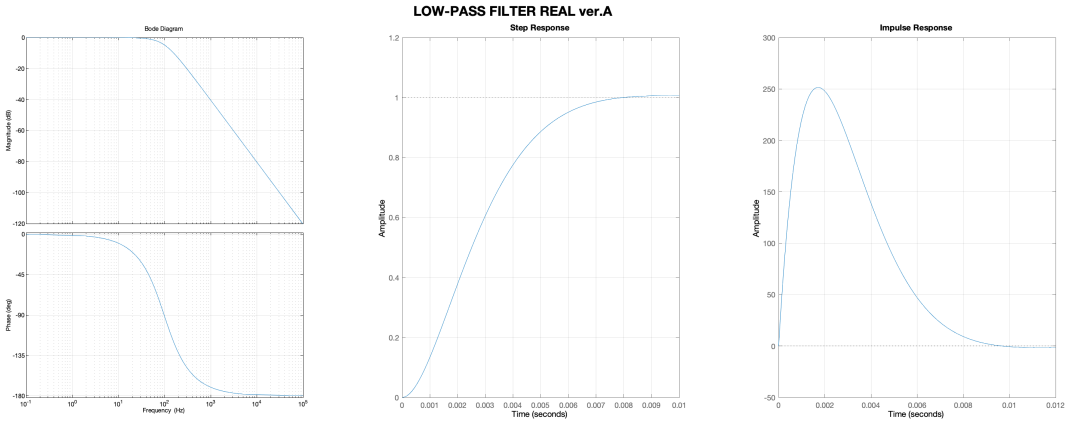


Figure 3.17. Low-Pass Filter Real vers.A Response.

MATLAB Code

```
%% Sallen_Key_Low_Pass_Filter (Second_Order_Bessel_Filter) [TI GUIDE]
R5_real = 18e3; %Ohm R5 = 1.7569e+04
R6_real = 10e3; %Ohm R6 = 10000
C9_real = 150e-09; %F C9 = 1.4418e-07
C10_real = 1.0000e-07; %F C10 = 1.0000e-07
f_LP_real = 1/(2*pi*sqrt(R5_real*R6_real*C9_real*C10_real))

G_LP_real = (1)/((s^2)*(R5_real*R6_real*C9_real*C10_real)+s*((R5_real*C10_real)+(
    R6_real*C10_real))+1);
options = bodeoptions;
options.FreqUnits = 'Hz';
options.Xlim = [0.1,100000];

figure('position', [2,742,2560,614])

subplot(1,3,1)
bode(G_LP_real, options);
grid on;

subplot(1,3,2)
step(G_LP_real);
grid on;

subplot(1,3,3)
impulse(G_LP_real);
grid on;

sgtitle('LOW-PASS FILTER REAL ver.A', 'fontweight', 'bold', 'fontsize', 18);
```

Low Pass Filter Version B

This type of circuit reported in Figure 3.18 contain inside the management of the K gain.

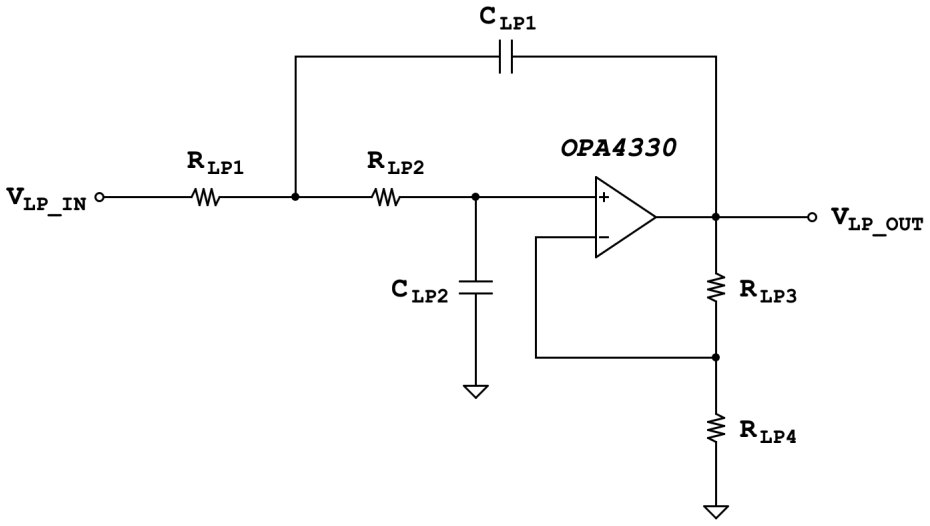


Figure 3.18. Low Pass Filter Version B Schematic.

Below are reported all the mathematical passages used to decide the values of the components that has been used in order to have a cut-off frequency wanted of:

$$f_{LP_{ideal}} = \frac{100Hz}{FSF} \tag{3.73}$$

It has been applied a system of equations to find the values of m and n similar to the one before studied but, with one more equation about gain K:

$$\begin{cases} K &= \dfrac{R_{LP4} + R_{LP3}}{R_{LP4}} \\ FSF \cdot f_{LP_{ideal}} &= \dfrac{1}{2 \cdot \pi \cdot R \cdot C \cdot \sqrt{mn}} \\ Q &= \dfrac{\sqrt{mn}}{m + 1 + mn \cdot (1 - K)} \end{cases} \tag{3.74}$$

It has been decided to have:

$$R = 330k\Omega \tag{3.75}$$
$$C = 47nF \tag{3.76}$$
$$K = 150 \tag{3.77}$$
$$R_{LP4} = 1k\Omega \tag{3.78}$$

What has been obtained is a second degree equation in which the coefficients are reported below:

$$an^2 + bn + c = 0 \tag{3.79}$$

where,

$$a = q^2 \cdot Q^2 - t^2 \tag{3.80}$$
$$b = 2 \cdot q \cdot Q^2 \tag{3.81}$$
$$c = Q^2 \tag{3.82}$$
$$t = FSF \cdot f_{LP_{ideal}} \cdot 2 \cdot \pi \cdot R \cdot C \tag{3.83}$$
$$q = 1 - K + t^2 \tag{3.84}$$

So, only one solution has been chosen:

$$n = 0.0141 \tag{3.85}$$
$$m = 0.7467 \tag{3.86}$$
$$R_{LP1_{ideal}} = 246.4k\Omega \tag{3.87}$$
$$R_{LP2_{ideal}} = 330k\Omega \tag{3.88}$$
$$C_{LP1_{ideal}} = 663pF \tag{3.89}$$
$$C_{LP2_{ideal}} = 47nF \tag{3.90}$$
$$R_{LP3_{ideal}} = 149k\Omega \tag{3.91}$$

The real values become:

$$R_{LP1_{real}} = 246k\Omega \tag{3.92}$$
$$R_{LP2_{real}} = 330k\Omega \tag{3.93}$$
$$C_{LP1_{real}} = 680pF \tag{3.94}$$
$$C_{LP2_{real}} = 47nF \tag{3.95}$$
$$R_{LP3_{real}} = 149k\Omega \tag{3.96}$$
$$f_{LP_{real}} = 98.81Hz \tag{3.97}$$

The transfer function of the filter in Figure 3.18 is shown below:

$$H_{LP}(s) = \frac{\dfrac{R_{LP4} + R_{LP3}}{R_{LP4}}}{s^2 \cdot (mR^2 \cdot nC^2) + s \cdot \left(mR \cdot C + R \cdot C - mR \cdot nC \cdot \left(\dfrac{R_{LP3}}{R_{LP4}}\right)\right) + 1} \tag{3.98}$$

At this point, after reporting the formulas in a MATLAB script, it has some tests in order to evaluate the filter's quality as in previous cases.

- **Ideal Low-Pass Filter Vers.B Result and MATLAB Code**
  In this subsection the ideal MATLAB results obtained and the code used have been presented.

  MATLAB Results

$f_{LP_{ideal}}$[Hz]	$R_{LP1_{ideal}}$[kΩ]	$R_{LP2_{ideal}}$[kΩ]	$C_{LP1_{ideal}}$[pF]	$C_{LP2_{ideal}}$[nF]
100	246.4	330	663	47

$R_{LP3_{ideal}}$[kΩ]	R_{LP4}[kΩ]
149	1

Table 3.7. Real values used during this simulation.

MATLAB Simulation

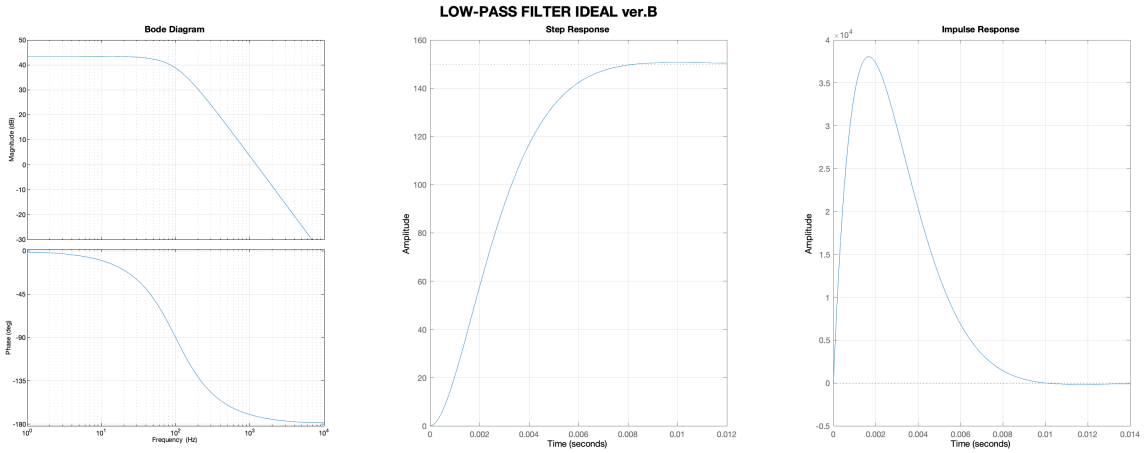


Figure 3.19. Low-Pass Filter Ideal vers.B Response.

MATLAB Code

```
%% Sallen_Key_Low_Pass_Filter (Second_Order_Bessel_Filter) [TI GUIDE]
FSF_LP = 1.2736; %1
Q_LP = 0.5773; %1
f_LP = 100/FSF_LP; %Hz
C_LP = 47e-9; %F
R_LP = 330e3; %Ohm
K_LP = 150; %1
R13 = 1e3; %Ohm

R12 = K_LP*R13 - R13

t = (FSF_LP*f_LP*2*pi*R_LP*C_LP);
q = (1-K_LP+t^2);

a = q^2*Q_LP^2-t^2;
b = 2*q*Q_LP^2;
c = Q_LP^2;

discriminante = b^2 - 4 * a * c;

if (discriminante >=0)
    if (discriminante == 0)
        n1 = -b / (2*a);
        disp("Soluzione reale equazione: n1= ");
        disp(n1);
```

```
    else
        n1 = (- b + sqrt(discriminante))/(2*a);
        n2 = (- b - sqrt(discriminante))/(2*a);
        disp("-> Soluzione reale equazione: n1=");
        disp(n1);
        disp("-> Soluzione reale equazione: n2=");
        disp(n2);
    end
else
    disp(" **** ERRORE - Soluzioni complesse !!!");
end

if (n1 > 0)
    m1 = 1/(((FSF_LP*f_LP*2*pi*R_LP*C_LP)^2)*n1);
    disp("-> Soluzione equazione: m1=");
    disp(m1);
    R10 = m1*R_LP
    R11 = R_LP
    C14 = n1*C_LP
    C15 = C_LP
    f_LP_idea1_1 = 1/(FSF_LP*2*pi*C_LP*R_LP*sqrt(m1*n1))
else
    disp(" **** ERRORE - Non esiste m1 !!! ****");
    disp(" ");
end

if (n2 > 0)
    m2 = 1/(((FSF_LP*f_LP*2*pi*R_LP*C_LP)^2)*n2);
    disp("-> Soluzione equazione: m2=");
    disp(m2);
    R10 = m2*R_LP
    R11 = R_LP
    C14 = n2*C_LP
    C15 = C_LP
    f_LP_ideal_2 = 1/(FSF_LP*2*pi*C_LP*R_LP*sqrt(m2*n2))

else
    disp(" **** ERRORE - Non esiste m2 !!! ****");
    disp(" ");
end

s = tf('s');
G_LP = ((R13+R12)/R13)/((s^2)*(R10*R11*C14*C15)+s*((R10*C15)+(R11*C15)+(R10*C14*(-(R12/
    R13))))+1);
options = bodeoptions;
options.FreqUnits = 'Hz';
options.Xlim = [1,10000];

figure('position', [2,742,2560,614],'NumberTitle', 'off', 'Name', 'LOW-PASS FILTER
    Ideal')

subplot(1,3,1)
bode(G_LP, options);
title('Bode Diagram', 'fontweight', 'bold', 'fontsize', 11);
grid on;

subplot(1,3,2)
step(G_LP);
```

```
82 grid on;
83
84 subplot(1,3,3)
85 impulse(G_LP);
86 grid on;
87
88 sgtitle('LOW-PASS FILTER IDEAL ver.B', 'fontweight', 'bold', 'fontsize', 18);
```

- **Real Low-Pass Filter Vers.B Result and MATLAB Code**
  In this subsection the real MATLAB results obtained and the code used have been presented. This result are the ones that come closest to reality, also because real values of resistors and capacitors have tolerances that make the real results differ from those calculated in these tests.

  MATLAB Results

$f_{LP_{real}}$[Hz]	$R_{LP1_{real}}$[kΩ]	$R_{LP2_{real}}$[kΩ]	$C_{LP1_{real}}$[pF]	$C_{LP2_{real}}$[nF]
98.81	246	330	680	47

$R_{LP3_{real}}$[kΩ]	R_{LP4}[kΩ]
149	1

Table 3.8. Real values used during this simulation.

MATLAB Simulation

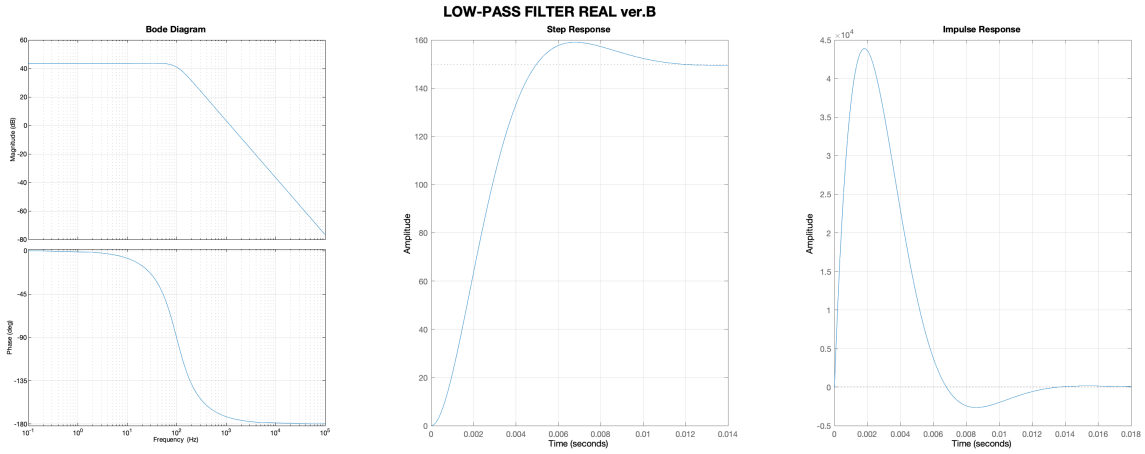


Figure 3.20. Low-Pass Filter Real vers.b Response.

MATLAB Code

```
%% Sallen_Key_Low_Pass_Filter (Second_Order_Bessel_Filter) [TI GUIDE]
R13_real = 1e3; %Ohm
R12_real = 149e3; %Ohm R12 = 149e3;
R10_real = 246e3; %Ohm R10 = 2.4640e+05
R11_real = 330e3; %Ohm R11 = 330000
C14_real = 680e-12; %F C14 = 6.6280e-10
C15_real = 4.7e-08; %F C15 = 4.7000e-08
f_LP_real = 1/(2*pi*sqrt(R10_real*R11_real*C14_real*C15_real))

G_LP_real = ((R13_real+R12_real)/R13_real)/((s^2)*(R10_real*R11_real*C14_real*C15_real)
    +s*((R10_real*C15_real)+(R11_real*C15_real)+(R10_real*C14_real*(-(R12_real/
    R13_real))))+1);
options = bodeoptions;
options.FreqUnits = 'Hz';
options.Xlim = [0.1,100000];

figure('position', [2,742,2560,614],'NumberTitle', 'off', 'Name', 'LOW-PASS FILTER Real
     Component')

subplot(1,3,1)

bode(G_LP_real, options);
title('Bode Diagram', 'fontweight', 'bold', 'fontsize', 11);
grid on;

subplot(1,3,2)
step(G_LP_real);
grid on;

subplot(1,3,3)
impulse(G_LP_real);
grid on;

sgtitle('LOW-PASS FILTER REAL ver.B', 'fontweight', 'bold', 'fontsize', 18);
```

3.2 Software Filters

In this last section is shown the use of MATLAB for the design of the software filters that has been inserted in the app in order to clean and visualize the signals collected by the circuit. Hardware filters have been used to clean the ECG signal before its amplification otherwise also the noise would be amplified. Software filters are applied after the signal has been collected so they have been used only for remove disturbances that are not part of the wanted signal. Starting from the acquired signal (an example is reported in the Figure 3.21), it has been noted that there are yet some problems: base line drift, bias and high-frequency disturbances.

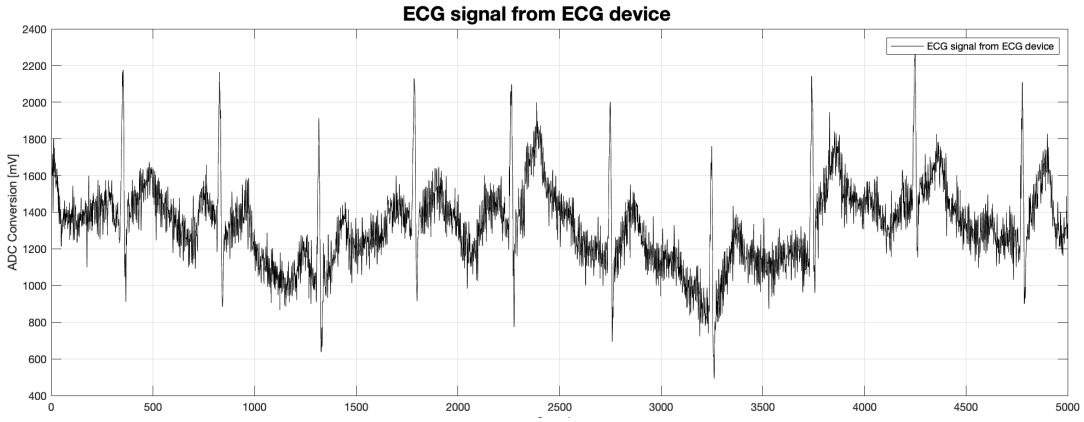


Figure 3.21. Acquired ECG signal from device.

3.2.1 Solution for base line drift and bias problem

Resolve bias problem is easy: with a function called "mean", the average of the signal is calculated and then subtracted from it. While a solution easy used to remove the base line drift is to subtract to the initial signal its baseline obtained through an High-Pass filter. The result is shown in Figure 3.22.

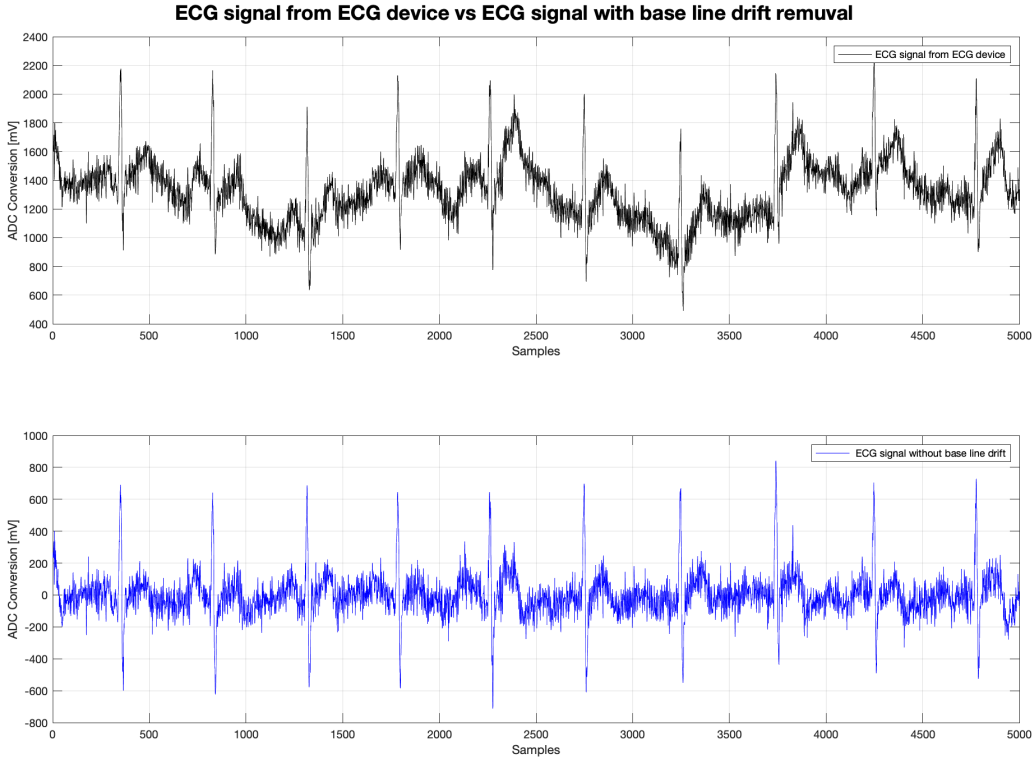


Figure 3.22. Acquired ECG signal from device.

MATLAB Code

```
d = fdesign.comb('notch', 'N,BW', 10, bw);
Hd = design(d,'SystemObject',true);

cofi = tf2sos(Hd.Numerator, Hd.Denominator)

ecg2 = sosfilt (cofi, ecgin-mean(ecgin));
```

3.2.2 Solution for high-frequency disturbances problem

In this case has been implemented a $50Hz$ notch filter in addition to second-order low-pass filter. The result is shown and in Figure 3.23 in Figure 3.24.

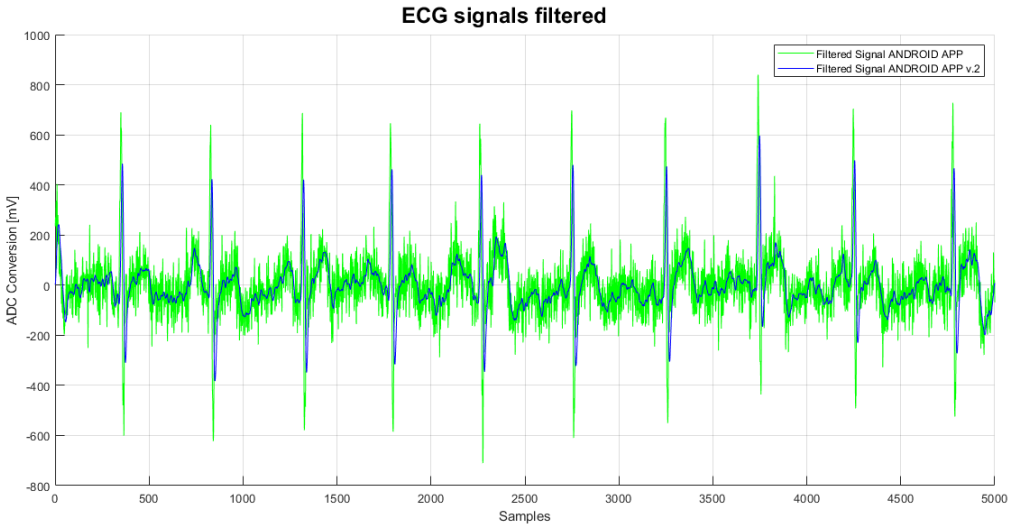


Figure 3.23. Filtered ECG signal.

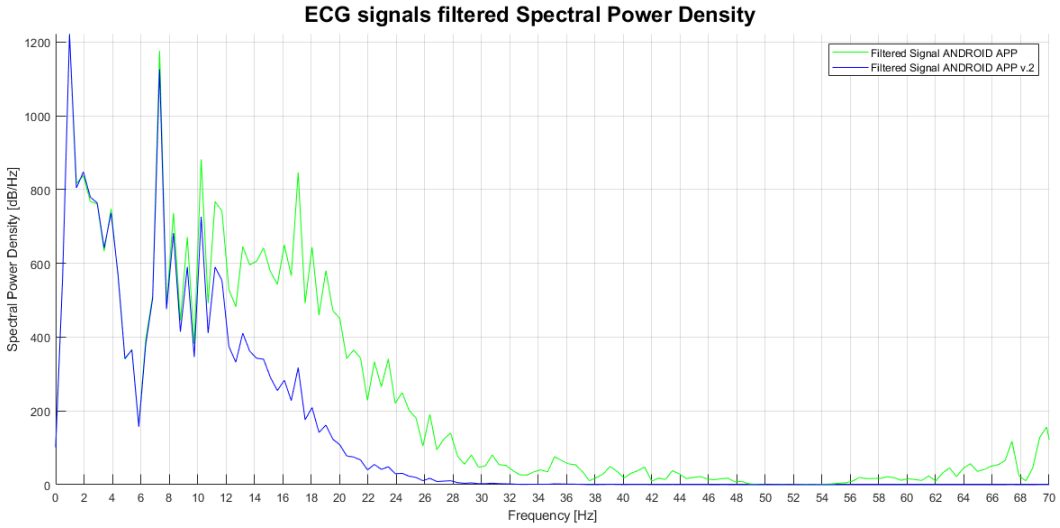


Figure 3.24. Spectral Power Density.

This last figure, that shows the Spectral Power density,is possible to see how the signal that interests us, that is, the one between the frequencies $0.5Hz$ and $50Hz$, remains while all the rest is filtered. There is a problem that must be resolved that is the the degrowth of the peaks due to an aggressive filtering.

MATLAB Code

```
clc;
clear all;
close all;

filename = 'Front_end.eml';

ECG = load_file(filename);

figure('position', [2,742,2560,614])
plot(ECG, 'k')
title('ECG signal from ECGX');
ylabel('ADC Conversion [mV]');
xlabel('Samples');
grid on;
hold on;
b = Motta_Filter_for_App(ECG, 10, 10, 10)';
c = Motta_Filter_for_App_2(b, 10, 10, 10)';

figure('position', [2,742,2560,614])
grid on;
hold on;
sgtitle('ECG signals filtered', 'fontweight', 'bold', 'fontsize', 18);
ylabel('ADC Conversion [mV]');
xlabel('Samples');
plot(b, 'g');
plot(c, 'bl');
legend('Filtered Signal ANDROID APP', 'Filtered Signal ANDROID APP v.2')

%% Spettrogramma
fs = 500;
t=0:1/fs:(length(b)-1)/fs;
b = b - (mean(b));
[Pxx2,f] = pwelch((b),hamming(1024),64,1024,fs);
c = c - (mean(c));
[Pxx3,f] = pwelch((c),hamming(1024),64,1024,fs);

figure('position', [0,42,2560,614])
hold on;
grid on;
sgtitle('ECG signals filtered Spectral Power Density', 'fontweight', 'bold', 'fontsize', 18)
    ;
ylabel('Spectral Power Density [dB/Hz]');
xlabel('Frequency [Hz]');
plot(f,Pxx2, 'g');
plot(f,Pxx3, 'bl');
legend('Filtered Signal ANDROID APP', 'Filtered Signal ANDROID APP v.2')
axis([0 70 0 (max(abs(Pxx2)))]);
xticks(0:2:70);
```

Functions MATLAB Code

- Motta_Filter_for_App

```
function [ecgout] = Motta_Filter_for_App( ecgin, n1, n2, n3 )

    if( nargin < 1 )
        return
    end
    if( nargin < 3)
        n1 = 5;
    end
    if( nargin < 4)
        n2 = 10;
    end
    if( nargin < 5)
        n3 = 15;
    end
    if( nargin > 5 )
        return
    end

    cofi = [ 0.8633 0.0000 -0.8633 1.0000 0.0000 -0.9381;
                1.0000 0.6180 1.0000 1.0000 0.5986 0.9381;
                1.0000 -0.6180 1.0000 1.0000 -0.5986 0.9381;
                1.0000 -1.6180 1.0000 1.0000 -1.5672 0.9381;
                1.0000 1.6180 1.0000 1.0000 1.5672 0.9381];

    ecg2 = sosfilt( cofi, ecgin - mean(ecgin) );

    ecgout = ecg2;

end
```

- Motta_Filter_for_App_2

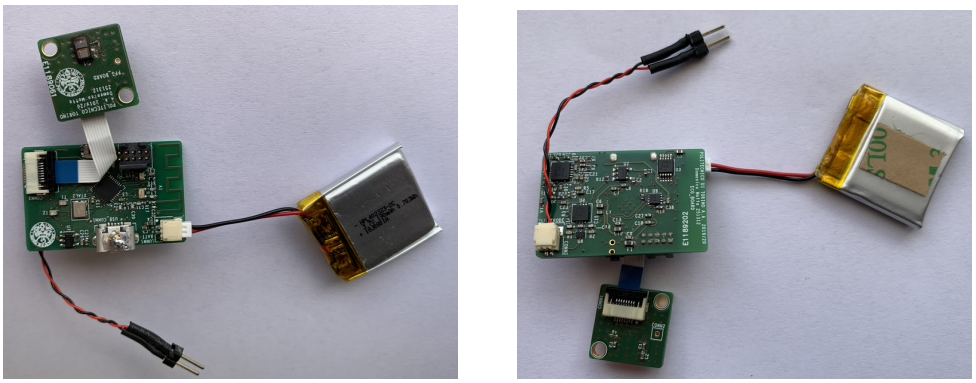
```
function [ecgout] = Motta_Filter_for_App_2( ecgin, n1, n2, n3 )

    if( nargin < 1 )
        return
    end
    if( nargin < 3)
        n1 = 5;
    end
    if( nargin < 4)
        n2 = 10;
    end
    if( nargin < 5)
        n3 = 15;
    end
    if( nargin > 5 )
        return
    end

    cofi = [0.007820208033497193 0.015640416066994387 0.007820208033497193
    1.000000000000000000 -1.734725768809274980 0.766006600943263893]; %% Fc = 15
         Hz, poles = 2

    ecg2 = sosfilt( cofi, ecgin - mean(ecgin) );

    ecgout = ecg2;

end
```

For implementing this filters on the app filter (IIR filter built with product of biquadratic's transfer functions) coefficients "cofi" have been calculated.

Chapter 4

Hardware

In this chapter has been introduce all passages used to build a final working PCB. Starting from the component choice, passing through the schematic presentation, PCB design explanation and the final circuit mounting. In the Figure 4.3 is shown the final circuit obtained.



(a) Top (b) Bottom

Figure 4.1. Final Circuit.

4.1 Components Choice

In this section has been explained the choice of component used to build the two PCBs one for the PPG sensor and the other one that contain the microcontroller, digital and analog power domain and the ECG acquisition. Below is shown the block diagram of the project.

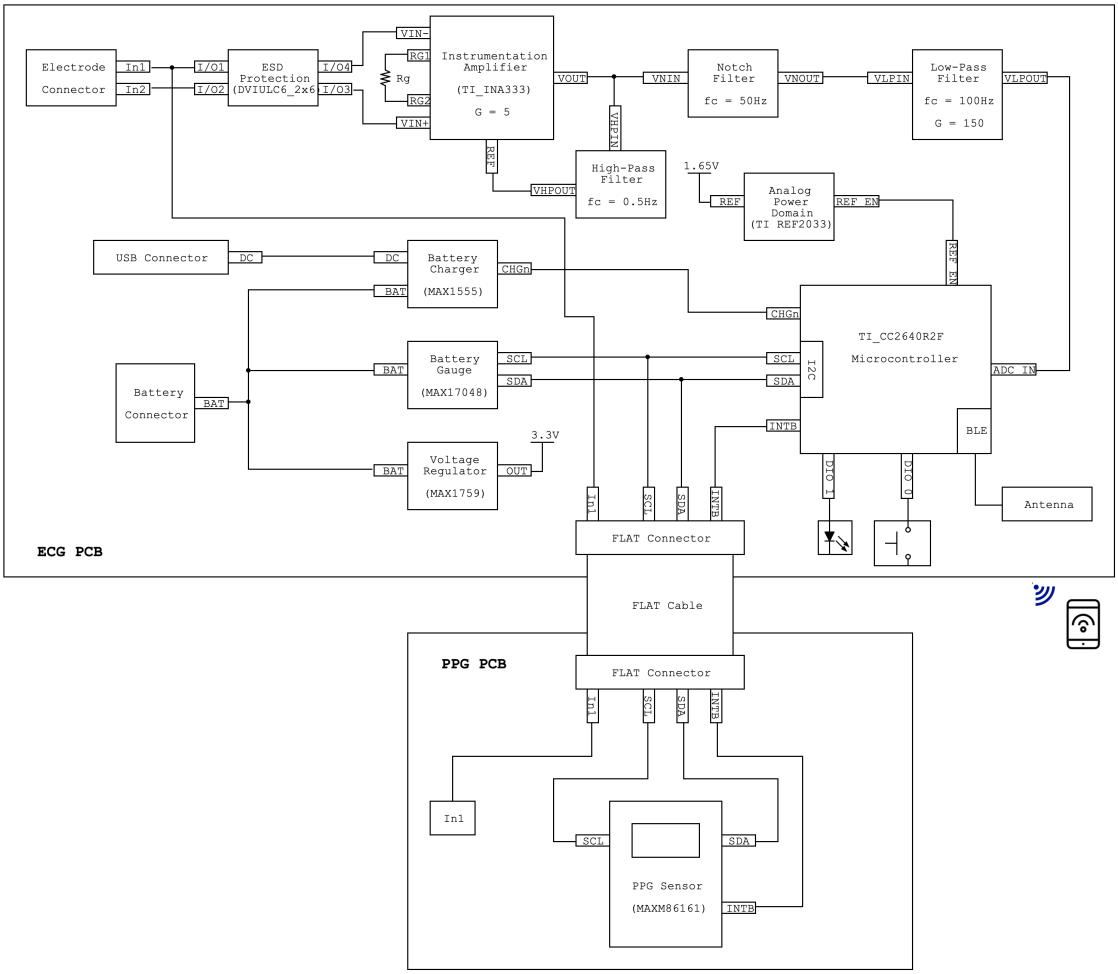


Figure 4.2. Circuit Block Diagram.

It is composed by two big blocks representing the two PCBs of which the circuit is formed.

- **ECG PCB Block:** is composed of:
  - *Connectors*: Electrodes Connector, USB Connector, Flat Connector and Battery Connector form the interaction between PCB and the outside world.
  - *Power Blocks*: Battery Charger, Battery Gauge are used to control the battery state and its charging mode; Voltage Regulator and Analog Power Domain are two components that generate stable power voltage and reference voltage used by all the other component that must have a stable power supply.
  - *Filters*: this block are used for ECG signal conditioning.
  - *Microcontroller*: control all other block and is used to collect data from ECG through its internal ADC Converter and Battery monitor and PPG value through I^2C communication.
- **PPG PCB Block:** is composed of two other main blocks:
  - *Flat Connector*: used to connect PPG PCB to ECG PCB through the flat cable that carrying the power lines and I2C lines.
  - *PPG Sensor (MAXM86161)*: sensor used to collect data about SpO_2 values.

4.1.1 Microcontroller TI CC2640R2F RSM

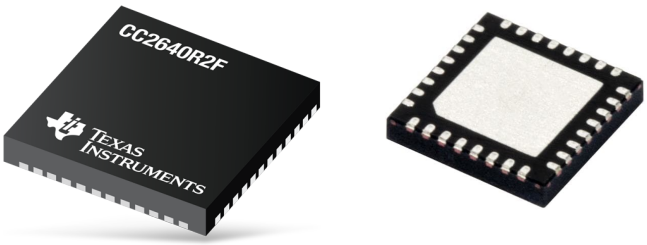


Figure 4.3. Microcontroller CC2640R2F Layout. [6]

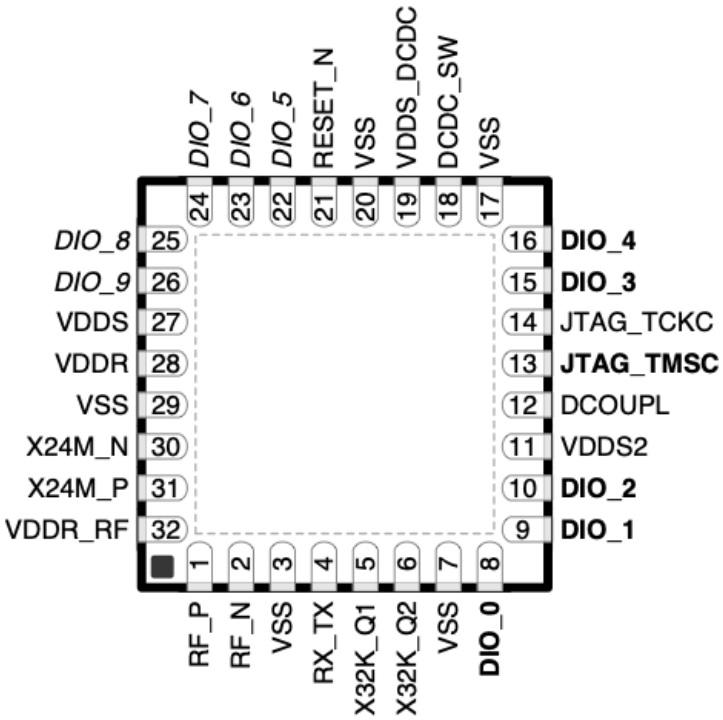


Figure 4.4. CC2640R2F RSM Pinout. [6]

This microcontroller has been chosen because of its little dimension with a discreet number of DIO pin.

Features:

- ARM®Cortex®-M3;
- Up to 48 MHz Clock Speed;
- Memories:
  - 275KB of Nonvolatile Memory Including 128KB of Programmable Flash;
  - Up to 28KB of System SRAM;
  - 8KB of SRAM Cache;
- RoHS-Compliant Packages:
  - 2.7mmx2.7mm YFV DSBGA34 (14 GPIOs);
  - 4mmx4mm RSM VQFN32 (10 GPIOs) chosed;
  - 5mmx5mm RHB VQFN32 (15 GPIOs);
  - 7mmx7mm RGZ VQFN48 (31 GPIOs);
- SPI, I^{2C} and I^2S Compatible;
- 2-Pin cJTAG and JTAG Debugging;
- 2.4-GHz RF Transceiver Compatible with Bluetooth Low Energy (BLE) 4.2;
- 1.8 to 3.8 V Power Supply;
- Sensor Controller 16 bit Coprocessor.

The CC2640R2F device is a wireless microcontroller (MCU) targeting Bluetooth®4.2 and Bluetooth®5 low energy applications.
The device is a member of the SimpleLink™ultra-low power CC26xx family of cost-effective, $2.4GHz$ RF devices. Very low active RF and MCU current and low-power mode current consumption provide excellent battery lifetime and allow for operation on small coin cell batteries and in energy-harvesting applications.
The SimpleLink Bluetooth low energy CC2640R2F device contains a 32-bit

ARM®Cortex®-M3 core that runs at 48 MHz as the main processor and a rich peripheral feature set that includes a unique ultra-low power sensor controller. This sensor controller is ideal for interfacing external sensors and for collecting analog and digital data autonomously while the rest of the system is in sleep mode. Thus, the CC2640R2F device is great for a wide range of applications where long battery lifetime, small form factor, and ease of use is important. Below is reported the Block diagram of the component: This

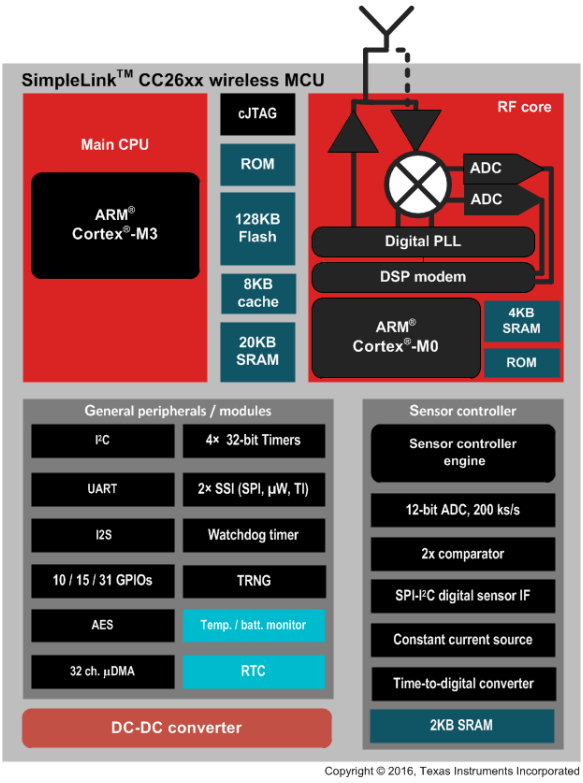


Figure 4.5. CC2640R2F RSM Block Diagram. [6]

is the core of the device, so its functions are:

- communicating through I^2C with MAX17048 in order to transfer data about the State of Charge and Voltage of the battery;
- communicating through I^2C in order to transfer data about Red Led and IR Led values collected by MAXM86161;
- acquiring data with the internal ADC from the ECG analog front-end output;

- communicating correctly with the internal Sensor Controller Coprocessor;
- sending and receiving data with a smartphone or tablet with BLE communication through a patch antenna.

There are three RF front-end configuration option recommended by Texas Instrument display in the Figure 4.6 and it has been chosen the single-ended in the middle.

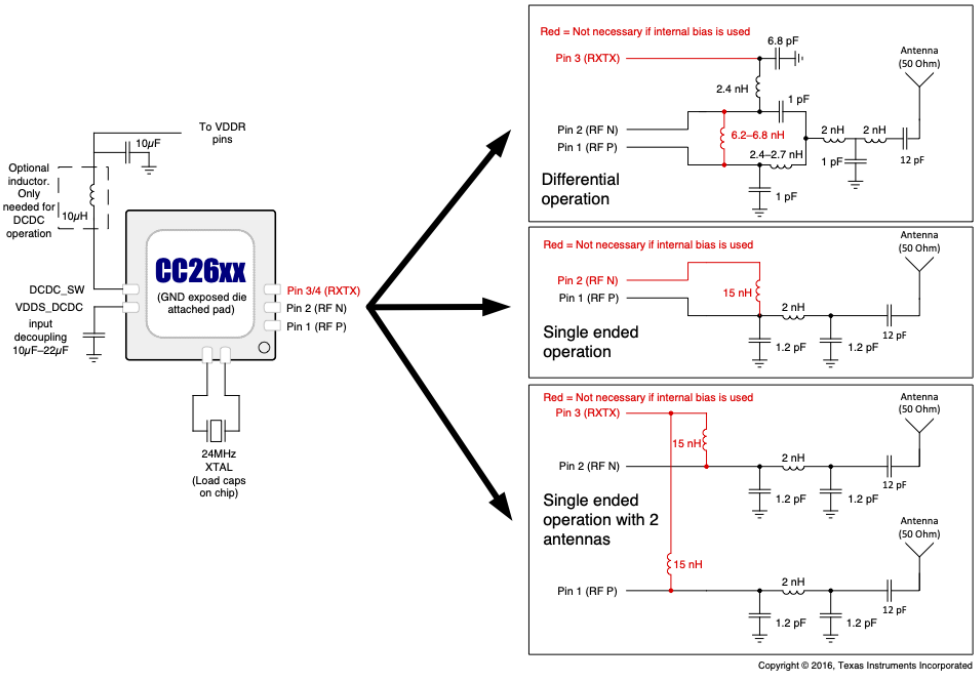


Figure 4.6. CC2640R2F RSM Front-end Antenna possibility. [6]

4.1.2 Antenna AN043

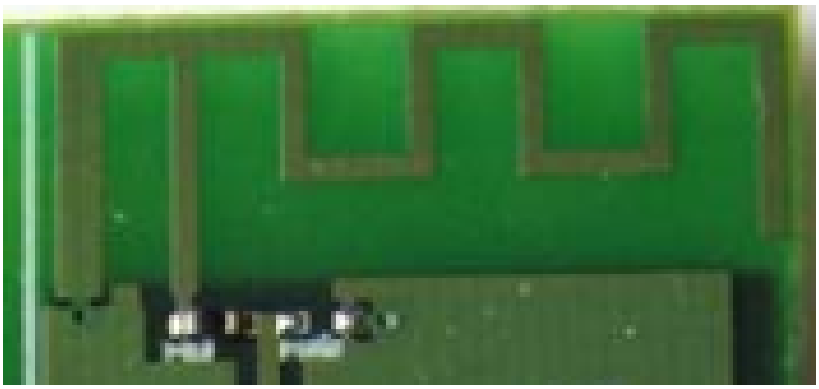
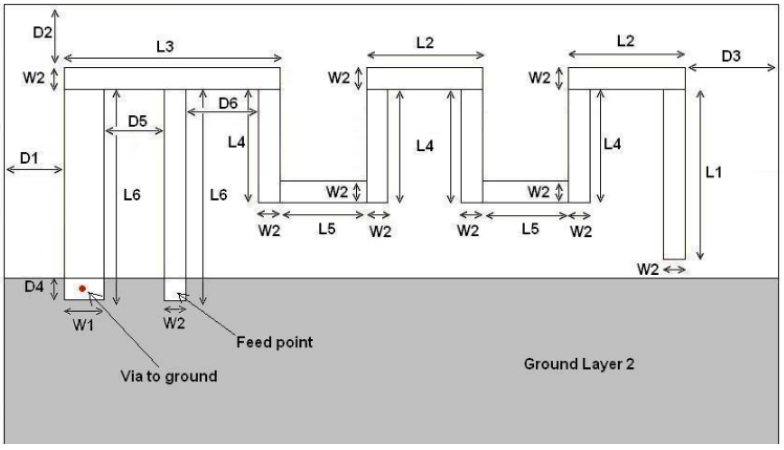


Figure 4.7. Antenna AN043.

The PCB antenna AN043 is a meandered Inverted F Antenna (IFA), designed to match an impedance of 50Ω at $2.45GHz$. Thus no additional matching components are necessary.



L1	3.94 mm
L2	2.70 mm
L3	5.00 mm
L4	2.64 mm
L5	2.00 mm
L6	4.90 mm
W1	0.90 mm
W2	0.50 mm
D1	0.50 mm
D2	0.30 mm
D3	0.30 mm
D4	0.50 mm
D5	1.40mm
D6	1.70 mm

Figure 4.8. Antenna AN043 dimensions.

The AN043 is one of the recommended PCB antennas by Texas Instruments. Filling a large part of the PCB, this antenna is the best choice among the possibilities, since it is the smallest one. Moreover, being a PCB antenna, it avoids additional cost for component purchase.

4.1.3 Voltage Regulators

MAX1759

The Voltage Regulator is used to generate a regulated output voltage from a single cell LiPo battery; the device operates over a wide $+1.6V$ to $+5.5V$ input voltage range and must generate a fixed $3.3V$ or adjustable ($2.5V$ to $5.5V$) output.

Figure 4.9. MAX1759. [7]

Features:

- Regulated Output Voltage (Fixed $3.3V$ or Adjustable $2.5V$ to $5.5V$);
- $100mA$ Guaranteed Output Current;
- $+1.6V$ to $+5.5V$ Input Voltage Range;
- $1\mu A$ Shutdown Mode;
- Load Disconnected from Input in Shutdown;
- Short-Circuit Protection and Thermal Shutdown;
- Small 10-Pin μMAX Package;

- *Electrical characteristics ($TA = 0°C$ to $+85°C$):*
  - Input Voltage Range: $1.6V$ to $5.5V$;
  - Output Voltage: $3.17V$ to $3.43V$;
  - Maximum Output Current: $100mA$;
  - C_X: $330nF$;
  - C_{IN}: $10\mu F$;
  - C_{OUT}: filter capacitor $10\mu F$;

Pin characteristics:

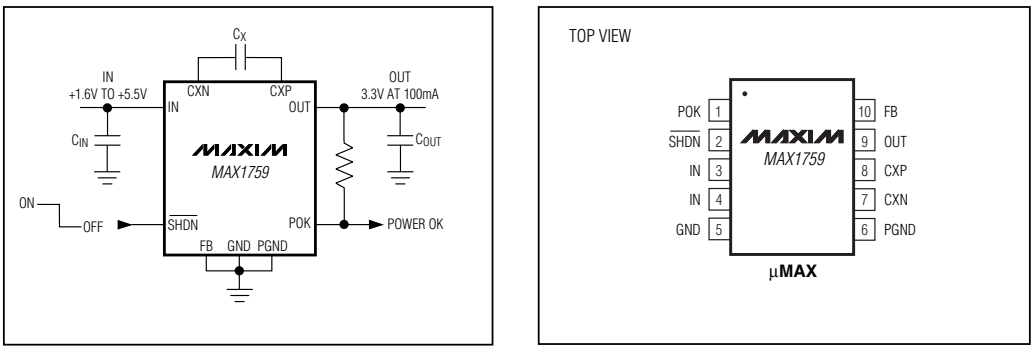


Figure 4.10. Pin Configuration and Typical Application Circuit. [7]

Applications:

- LiPo Battery-Powered Applications;
- Miniature Equipment;
- Backup Battery Boost Converters;
- Translators;

TI REF2033

This voltage regulator is used in many applications in which is required an additional stable voltage in the middle of the ADC input range to bias for example an input bipolar signals as in this project. This component is able to provide two stable voltages V_{REF} and V_{BIAS}

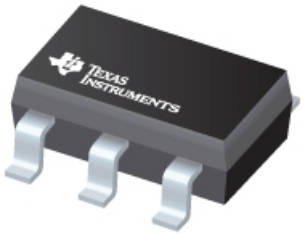


Figure 4.11. REF2033. [8]

Features:

- provides two Stable Voltage, V_{REF} and $V_{REF}/2$, that is suitable for use in single-supply voltage systems;
- High Initial Accuracy of $\pm 0.05\%$ (max);
- Small $SOT23-5$ Package;
- *Electrical characteristic:*
  - Low Dropout Voltage: $10mV$;
  - High Output Current: $\pm 20mA$;
  - Low Quiescent Current: $360\mu A$;
  - Line Regulation: $3ppm/V$;
  - Load Regulation: $8ppm/mA$.

Pin characteristics:

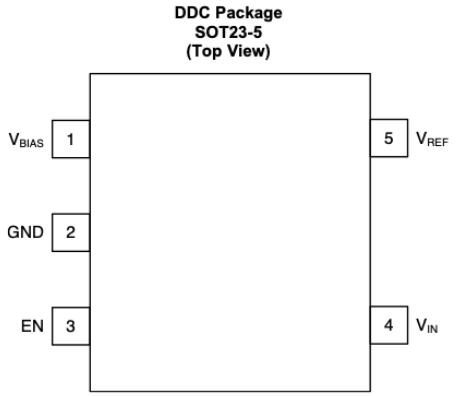


Figure 4.12. Pin Configuration. [8]

- V_{BIAS}: Output Voltage ($V_{REF}/2 = 1.65V$);
- GND: Ground;
- EN: Input Pin used to Enable the Device with ($EN \geq VIN - 0.7V$);
- V_{IN}: Input Supply Voltage;
- V_{REF}: Reference Output Voltage ($V_{REF} = 3.3V$).

Applications:

- Medical Equipment;
- Data Acquisition Systems;
- Single-Supply Systems.

4.1.4 Battery Charger MAX1555

The Battery Charger is used to charge a single-cell LiPo battery from both USB and AC adapter sources. It operates with no external FETs or diodes, and accept operating input voltages up to 7V.

Figure 4.13. MAX1555. [9]

Features:

- Charge from USB or AC Adapter;
- Automatic Switch over when AC Adapter is plugged IN;
- On-Chip Thermal Limiting Simplifies Board Design;
- Charge Status Indicator;
- 5-Pin Thin SOT23 Package;
- Electrical characteristics ($TA = 0°C$ to $+85°C$):
  - DC Voltage Range: $3.7V$ to $7.0V$;
  - USB Voltage Range: $3.7V$ to $6.0V$;
  - BAT Regulation Voltage: $4.158V$ to $4.242V$;
  - DC to BAT Voltage Range: $0.1V$ to $6.0V$;
  - $\overline{CHG}$, $\overline{POK}$ Logic-Low Output: $300mV$

Pin characteristics:

- USB: USB Port Charger Supply Input. USB draws up to $100mA$ to charge the battery. Decouple USB with a $1\mu F$ ceramic capacitor to GND;
- GND: Ground;
- $\overline{POK}$: Power-OK Active-Low Open-Drain Charger Status Indicator;
- $\overline{CHG}$: Active-Low Open-Drain Charge Status Indicator. $\overline{CHG}$ pulls low when the battery is charging. $\overline{CHG}$ goes to a high-impedance state, indicating the battery is fully charged, when the charger is in voltage mode and charge current falls below $50mA$;
- DC: DC Charger Supply Input for an AC Adapter. DC draws $280mA$ to charge the battery. Decouple DC with a $1\mu F$ ceramic capacitor to GND
- BAT: Battery Connection. Decouple BAT with a $1\mu F$ ceramic capacitor to GND.

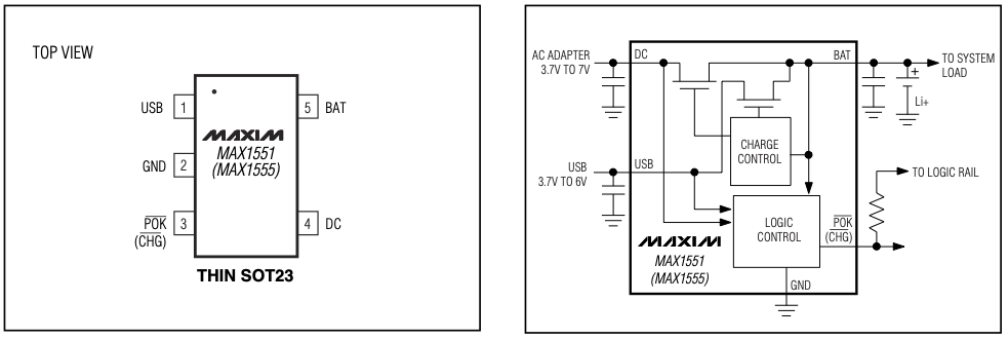


Figure 4.14. Pin Configuration and Typical Application Circuit. [9]

Applications:

- PDAs;
- Wireless Appliances;
- Cell Phones;
- Digital Cameras;

4.1.5 Battery Gauge MAX17048

Fuel Gauge device is used to track the battery relative state-of-charge (SOC) continuously over widely varying charge and discharge conditions.

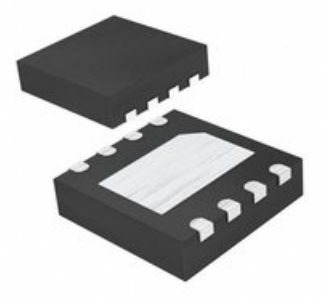


Figure 4.15. MAX17048. [10]

Features:

- Precision $\pm 7.5mV$/Cell Voltage Measurement;
- Used for 1 Cell LiPo Battery;
- I^2C Interface;
- 8-Bit OTP ID Register;
- Configurable Alert Indicator;
- Programmable Reset for Battery Swap $2.28V$ to $3.48V$ Range;
- Reports Charge and Discharge Rate;
- Battery-Insertion Debounce
- *Electrical characteristics* ($TA = -20°C$ *to* $+70°C$)*:*
  - Supply Voltage: $2.5V$ to $4.5V$;
  - Data I/O Pins: $-0.3V$ to $5.5V$;

- Maximum Output Current:
  - Sleep mode: $2\mu A$;
  - Hibernate mode: $5\mu A$;
  - Active mode: $40\mu A$;
- Voltage Error: $7mVxCell$;
- Voltage-Measurement Resolution: $1.25mVxCell$;
- SCL Clock Frequency: $400kHz$;

Pin characteristics:

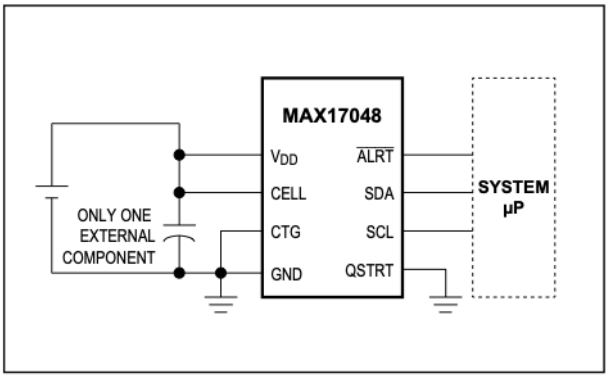


Figure 4.16. Typical Application Circuit. [10]

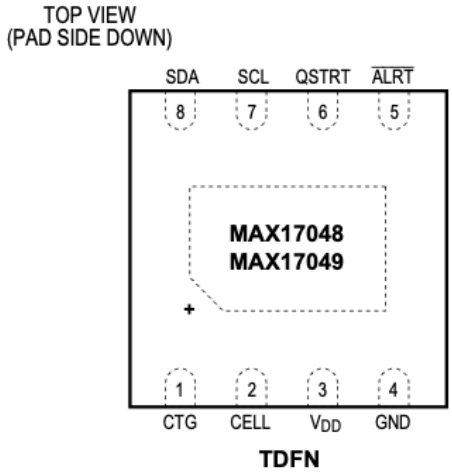


Figure 4.17. Pin Configuration. [10]

Applications:

- Smartphones, Tablets;
- Smartwatches, Wearables;
- Bluetooth Headsets;
- Health and Fitness Monitors;
- Digital Still, Video, and Action Cameras;
- Medical Devices;
- Handheld Computers and Terminals;
- Wireless Speakers;
- Home and Building Automation, Sensors;

4.1.6 ESD Protection

An ESD protection component is used to protects a circuit from Electrostatic discharge (ESD) that can cause malfunction or breakdown of electronic device.

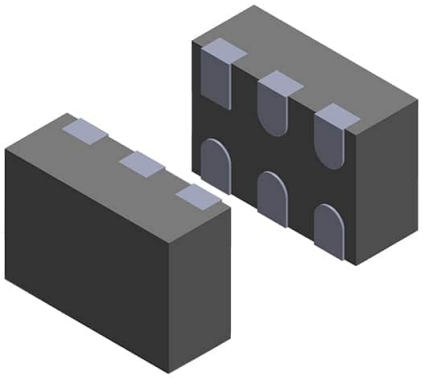


Figure 4.18. DVIULC6-2x6. [11]

Features:

- Two Lines ESD Protection (at $15kV$ air and contact discharge, exceeds $IEC61000-4-2$);
- Fast Response Time Compared with Varistors;
- RoHS compliant;
- Small Package $1.45mm^2$ for µQFN;
- *Electrical characteristics*
  - Ultra Low Capacitance: $0.6pF$ at $f = 825MHz$;
  - Low Leakage Current: $0.5\mu A$ max;

Pin characteristics:

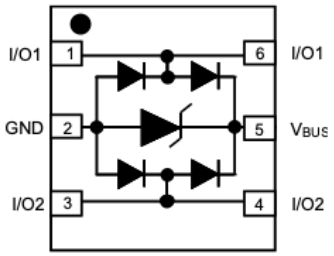


Figure 4.19. Pin Configuration. [11]

Applications:

- Medical Devices ESD Protection;

4.1.7 Instrumentation Amplifier TI INA333

TI INA333 is a low power instrumentation amplifier used in this project to initial amplify the ECG signal before filter it. Is possible to sets gain from 1 to 1000 according to the value of one single resistor R_G.

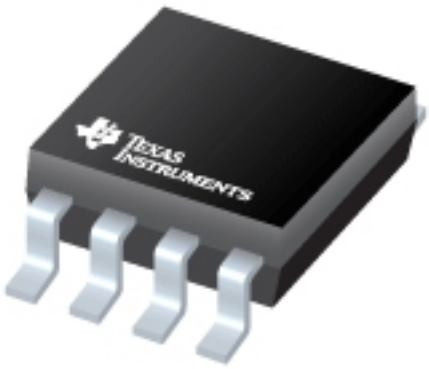


Figure 4.20. INA333. [12]

Features:

- 8-Pin VSSOP;
- *Electrical characteristics*
  - Supply Range: $1.8V$ to $5.5V$;
  - Input Voltage: (V–) $+0.1V$ to (V+) $-0.1V$;
  - Output Range: (V–) $+0.05V$ to (V+) $-0.05V$;
  - High CMRR: $100dB$ (Minimum), $G \geq 10$;
  - Low Quiescent Current: $50\mu A$;
  - Low Offset Voltage: $25\mu V$ (Maximum), $G \geq 100$;
  - Low Drift: $0.1\mu V/°C$, $G \geq 100$.

Pin characteristics:

- R_G: Gain setting pins; place a gain resistor between pins 1 and 8;
- $V+$: Positive supply voltage;
- $V-$: Negative supply voltage;
- V_{IN+}: Positive input;
- V_{IN+}: Negative input;
- REF: Reference input. This pin must be driven by low impedance or connected to Ground.;
- V_{OUT}: Output.

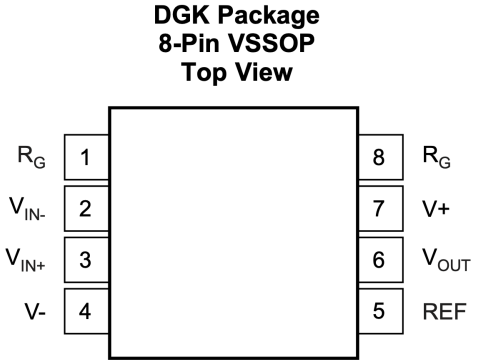


Figure 4.21. Pin Configuration. [12]

Applications:

- ECG Amplifiers;
- Medical Instrumentation;
- Portable Instrumentation;
- RTD Sensor Amplifiers;
- Data Acquisition;

4.1.8 Operational Amplifier TI OPA4330

The TI OPA4330 is a CMOS operational amplifier that is member of the Zero-Drift family of amplifiers and it can offer precision performance at a very low price. This is the reason why it has been chosen. The version chosen, contains 4 operational amplifiers.

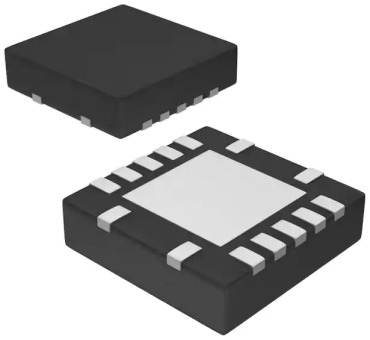


Figure 4.22. OPA4330 Component. [13]

Features:

- Packages: DSBGA, SC70, VQFN (chosen);
- Internal EMI Filtering;
- Rail-to-Rail Input and Output;
- *Electrical characteristics*
  - Supply Range: $1.8V$ to $5.5V$;
  - Low Noise: $1.1\mu VPP$, $0.1Hz$ to $10Hz$;
  - Low Quiescent Current: $35\mu A$ (Maximum);
  - Zero Drift: $0.25\mu V/°C$ (Maximum);
  - Low Offset Voltage: $50\mu V$ (Maximum).

Pin characteristics:

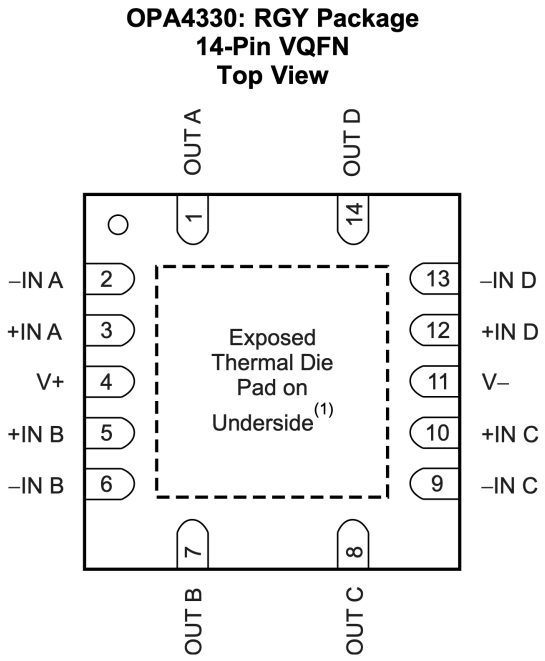


Figure 4.23. Pin Configuration. [13]

Applications:

- Medical Instrumentation.

4.1.9 PPG Sensor MAXM86161

The MAXM86161 is a low-power, integrated, optical data acquisition system.

Figure 4.24. MAXM86161 Component. [14]

Features:

- Built-In Algorithm for Rejection of Fast Ambient Transients;
- Heart Rate and SpO_2 Monitoring;
- High Resolution ADC (19 bit);
- Three 8-Bit LED Current DACs;
- Low-Power Operation for Wearable Devices;
- Package: 14-pin OLGA.

Pin characteristics and Internal Block Diagram:

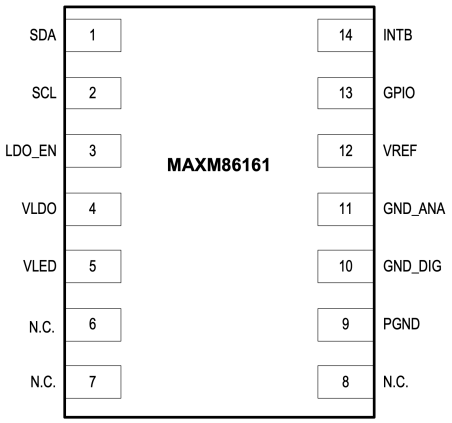


Figure 4.25. Pin Configuration. [14]

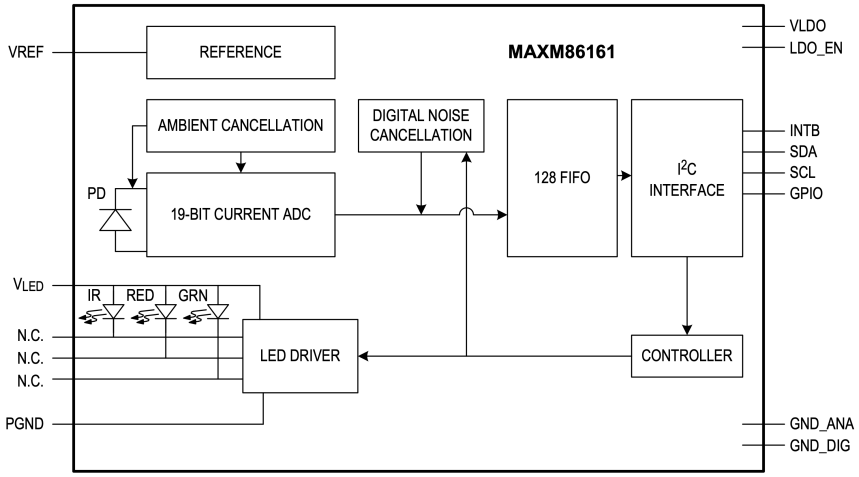


Figure 4.26. Internal Block Diagram. [14]

Applications:

- Medical Instrumentation.

4.1.10 Connectors, Button and LED

Flat Connector and Cable

Flat connectors are used to connect the two PCBs. They are soldered to PCBs and they are connected each other with a Flat Cable. The cable carry both signals and power lines. In this project are used the 8 positions/pins connectors.

Figure 4.27. Flat Connector. [15]

Figure 4.28. Flat Cable. [15]

Battery and ECG Connector

This is the JSTConnector header connector SMD two position 1MM SM02B-SRSS-TB(LF)(SN).

Figure 4.29. JST Connector Component. [15]

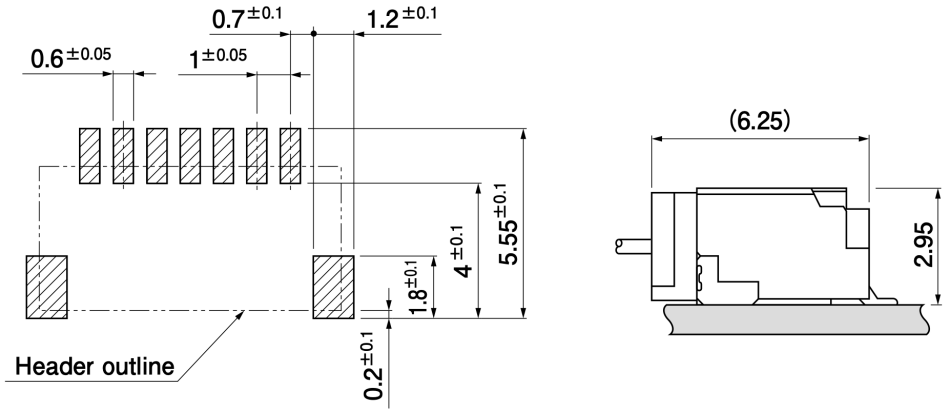


Figure 4.30. JST Connector Footprint. [15]

JTAG Connector

The JTAG Connector is used to connect the microcontroller to the PC in order to program the processor and also debug it.

Figure 4.31. JTAG Connetor. [18]

Red LED

The Red LED is used as indicator during acquisition. Flashes during ECG acquisition and Battery measurement.

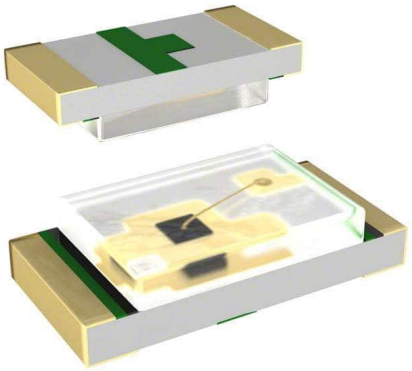


Figure 4.32. LED Red Component. [16]

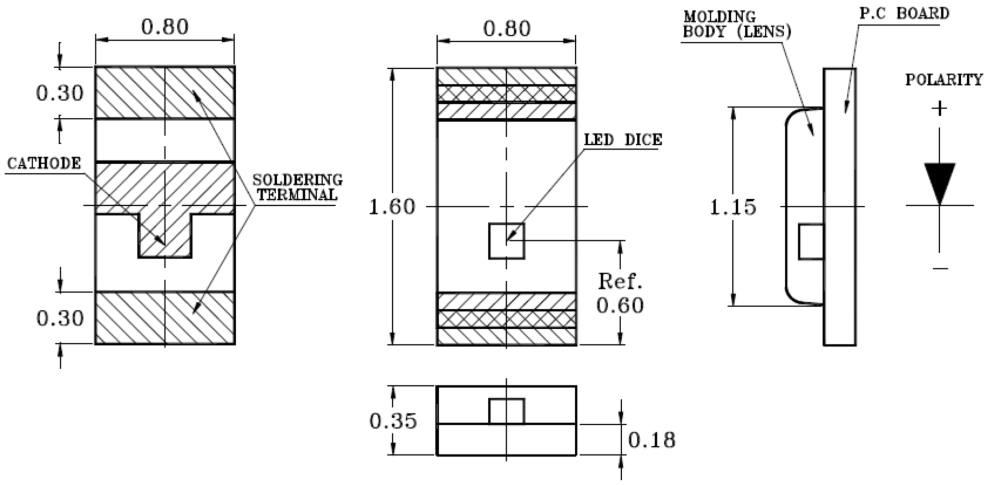


Figure 4.33. LED Red Footprint. [16]

Push Button

Not used in this project but, inserted for future perspectives.

Figure 4.34. Light Touch Switches Component. [17]

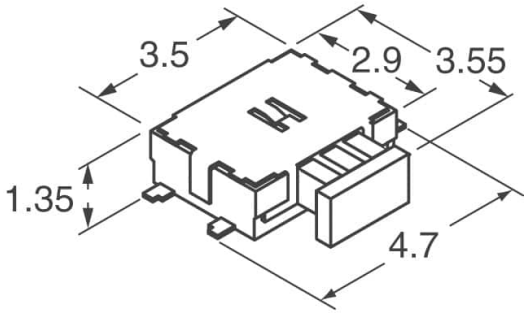


Figure 4.35. Light Touch Switches Dimension. [17]

4.2 Schematic Explanation

In this section have been presented all schematic circuit of this project and is divided in two subsections:

- **Test Boards Circuit**: in which has been shown the circuit used for tests. This is also divided in:
  - *Test Boards Circuit Version A*;
  - *Test Boards Circuit Version B*.
- **Final ECG and PPG Circuit**: in which has been shown and explained the final circuit. Divided in:
  - *ECG and PPG Circuit*: has been shown and explained all the blocks that form the core of the device and the simple peripheral circuit connect to the main one.

4.2.1 Test Boards Circuit

As mentioned in the previous chapters, before testing the final circuit, two test boards has been designed to test the various filter configurations studied. Two versions Version A and Version B have been designed and produced. As already reported in the MATLAB chapter, in the computer tests the circuit version B has been preferred, but in any case it has been wanted to test their functioning in reality. In these schematics, in addition to the front-end part, the power domain has also been included. This is because the boards have been connected to the PCB of the old ECG in order to have a functioning acquisition block (see more in chapter Tests). Below are the two schematics of the Front-End tested.

Test Boards Circuit Version A

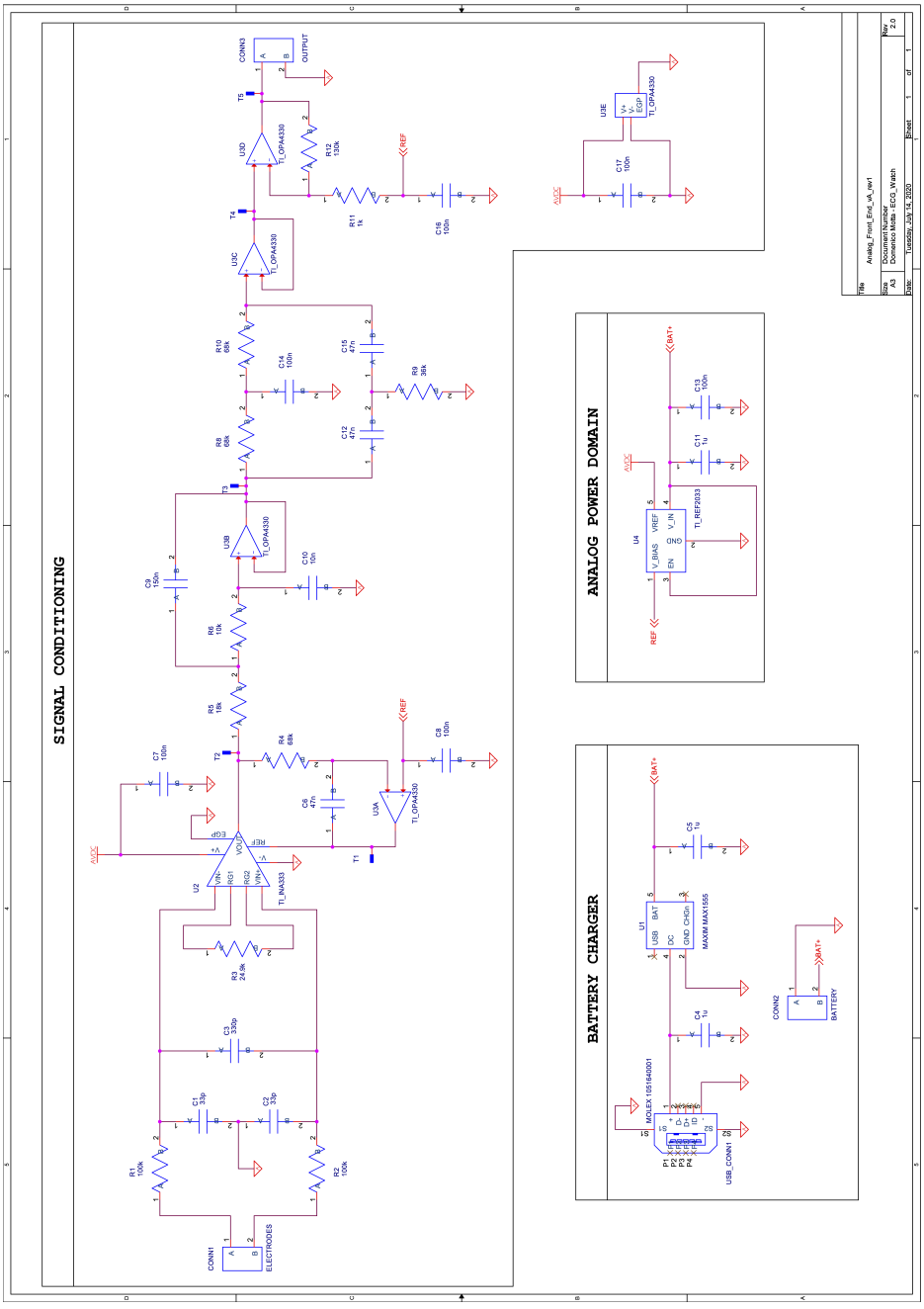


Figure 4.36. Test Boards Circuit Version A Schematic.

Test Boards Circuit Version B

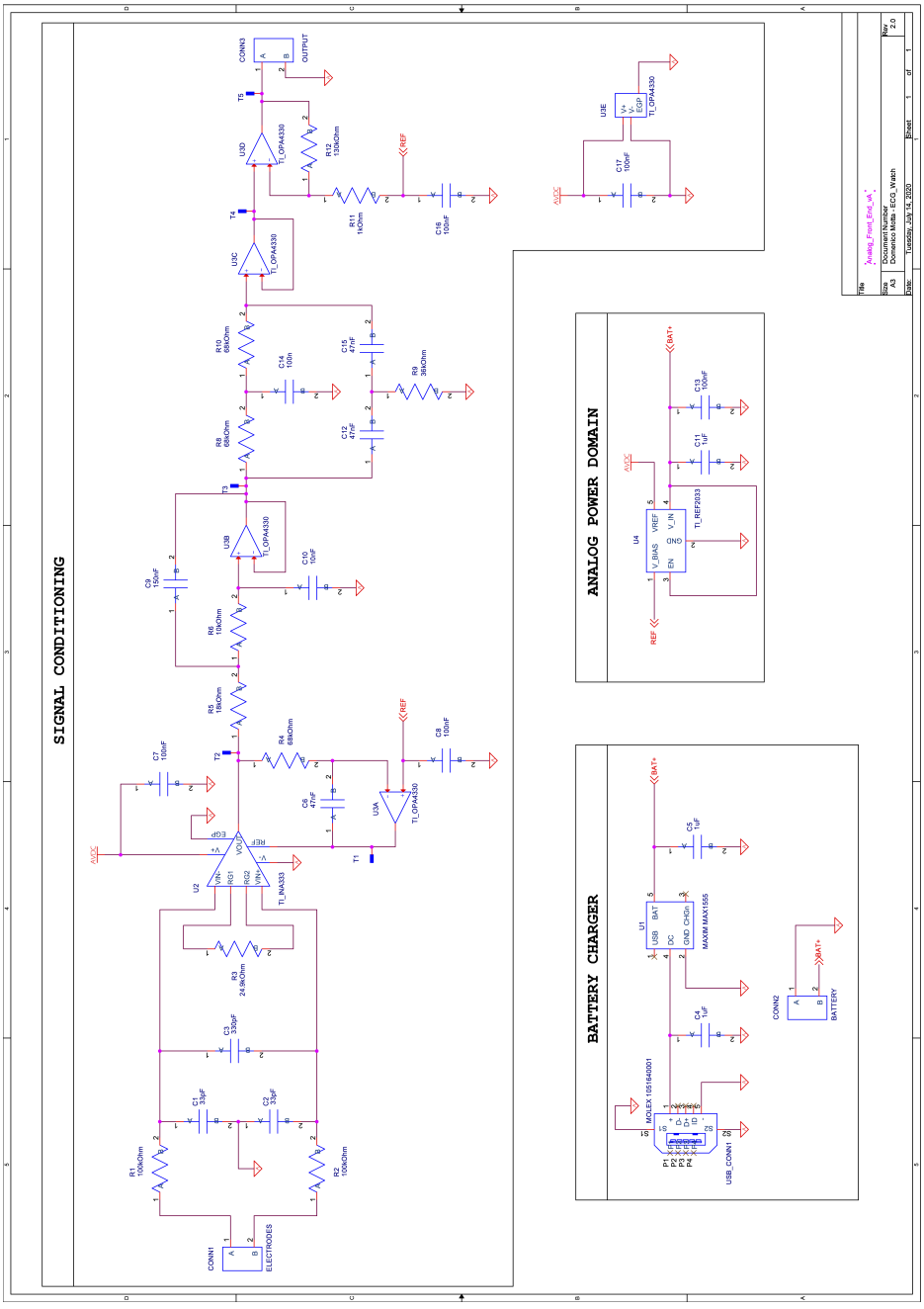


Figure 4.37. Test Boards Circuit Version B Schematic.

4.2.2 Final ECG and PPG Circuit

Front End Schematic

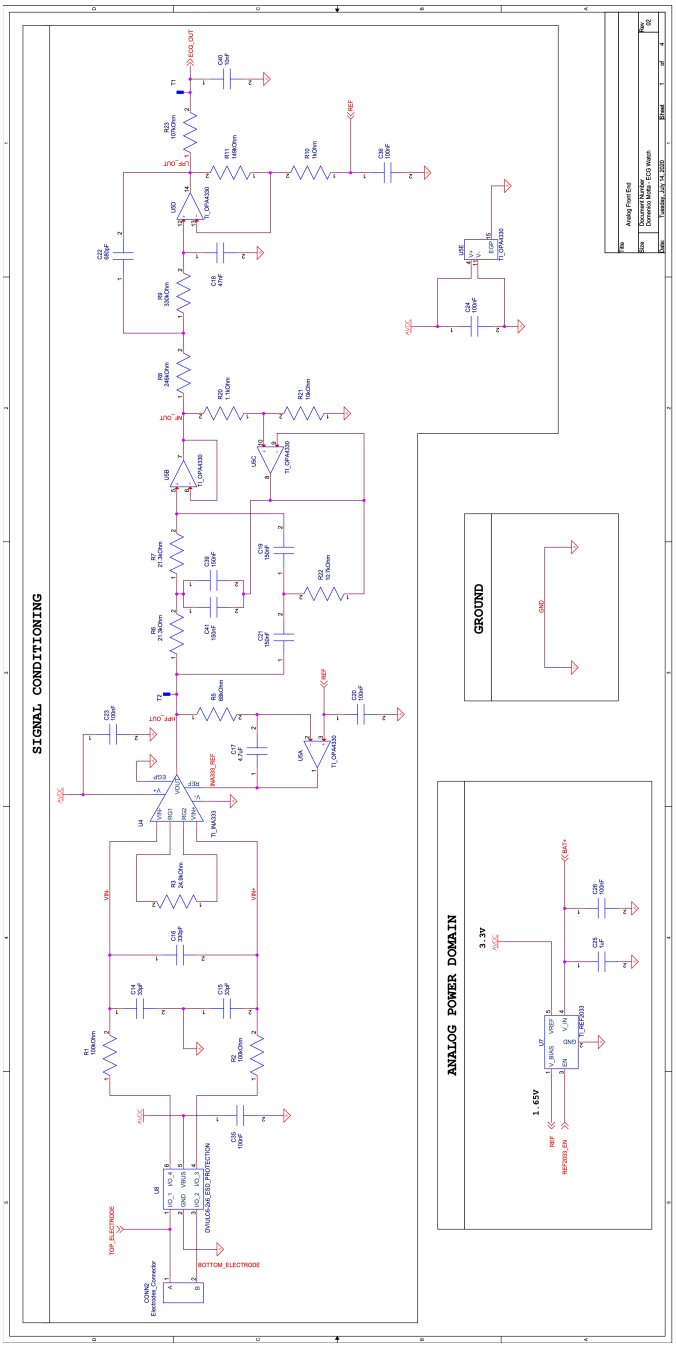


Figure 4.38. Analog Front End Schematic.

Simple explanation:
The front-end part has been explained exhaustively in the previous chapters. Focus on the analog power domain block used for the generation of two stable voltages useful for the filtering and amplification blocks.

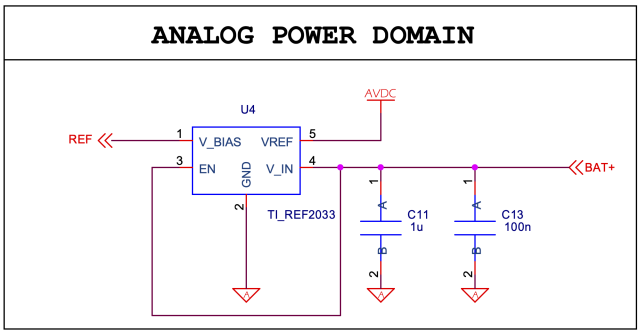


Figure 4.39. Analog Power Domain Schematic.

For doing that is used:

- $TIREF2033$ component;
- $C11$ and $C13$ are decoupling capacitor.

Power Management Schematic

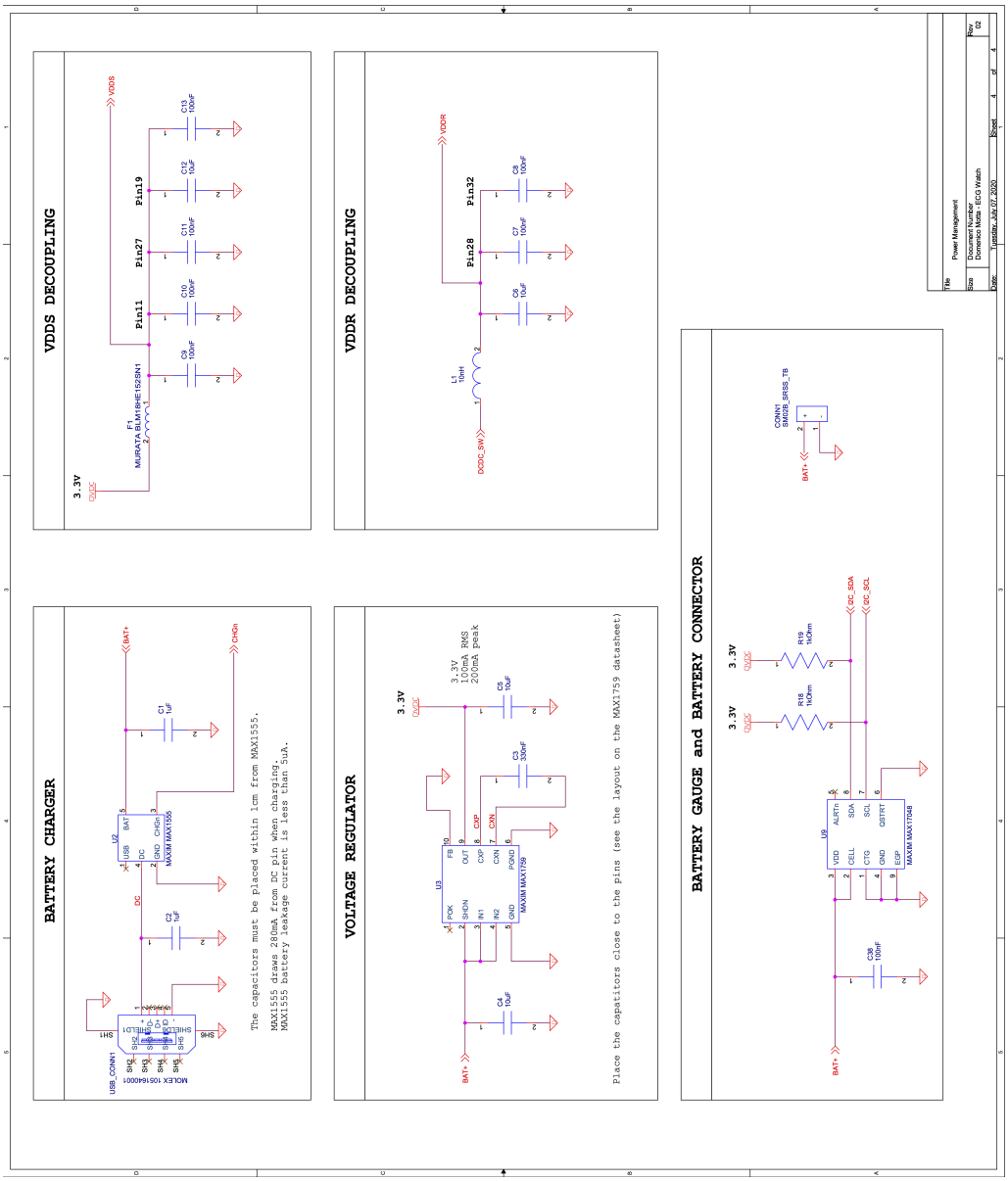


Figure 4.40. Power Management Schematic.

Simple explanation:

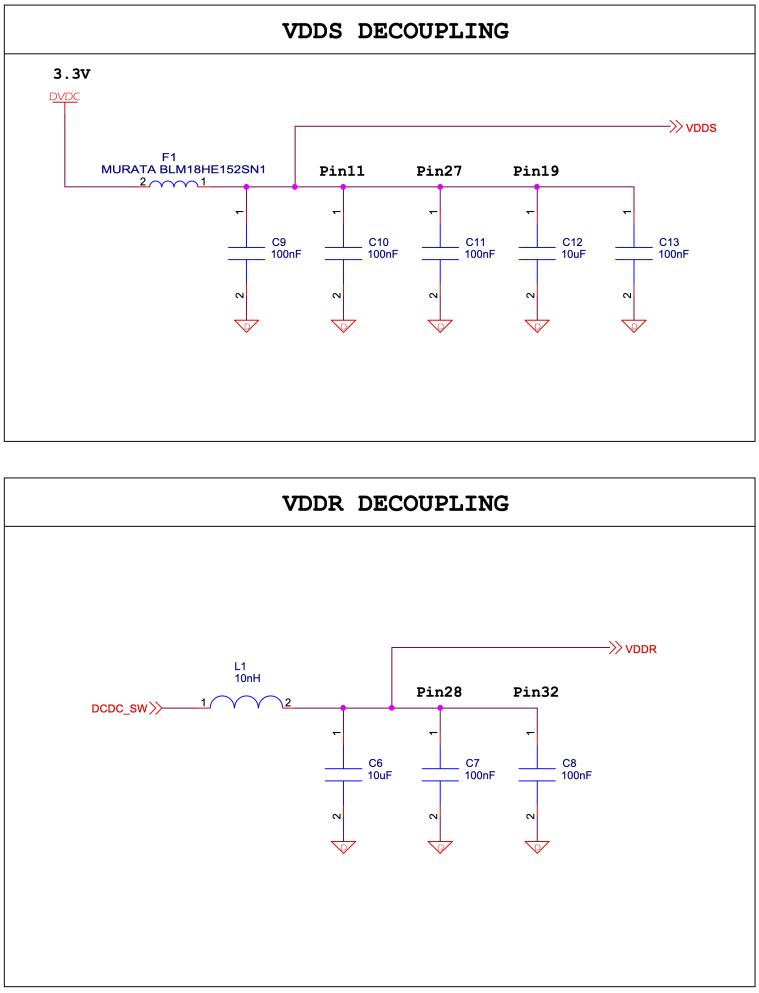


Figure 4.41. Decoupling Capacitors.

- VDDS decoupling:
  - $C9 \rightarrow C13$ are decoupling capacitors;
  - $F1$ is a ferrite for EMC compliance.
- VDDR decoupling:
  - $L1$ and $C6$ are components needed by the microcontroller to make the DCDC internal switching voltage regulator work better;
  - $C7$ and $C8$ are decoupling capacitors.

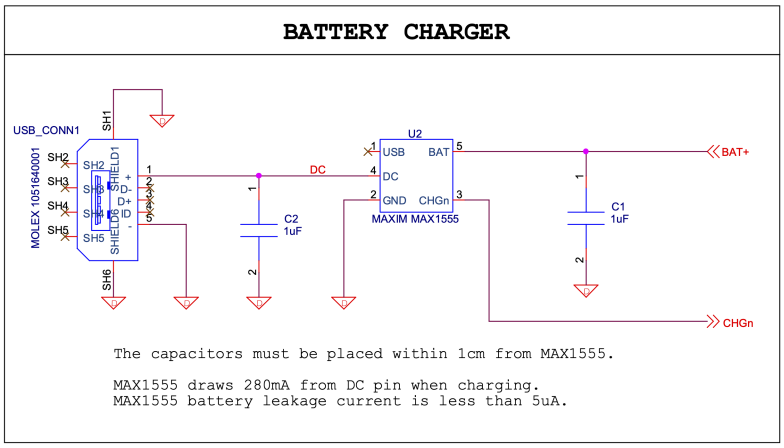


Figure 4.42. Battery Charger.

The $MAX1555$ is the component used as interface between USB connector and the Lipo Battery.

- $C1$ and $C2$ are decoupling capacitors.

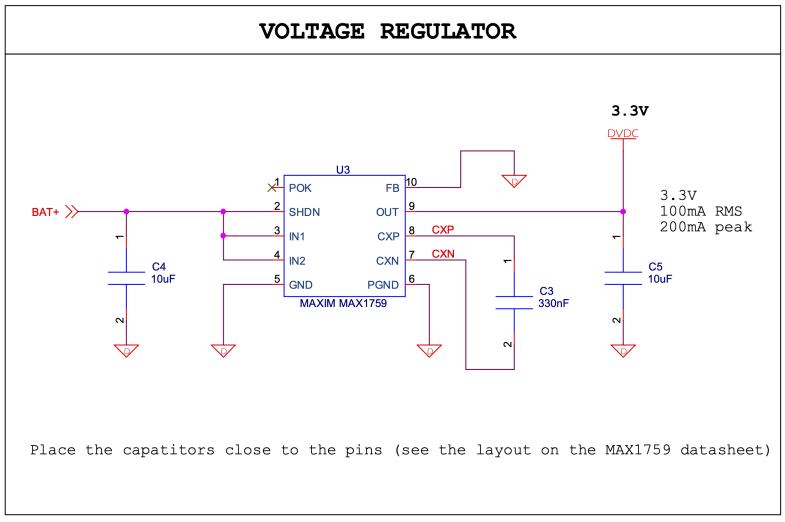


Figure 4.43. Voltage Regulator.

The $MAX17048$ is the component used to generate a supply voltage for the circuit.

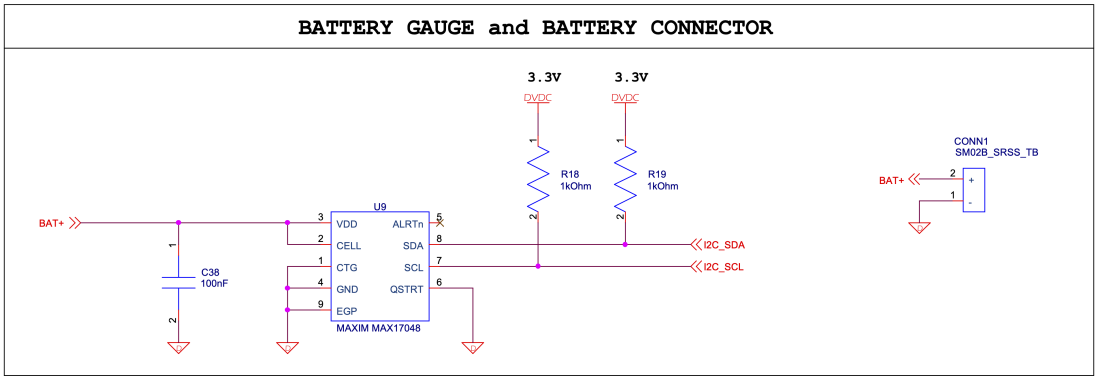


Figure 4.44. Battery Gauge.

The $MAX17048$ is the component used to measure the voltage of the Lipo battery and it State of Charge.

- $R18$ and $R19$ are the pull-up resistor for the I^2C communication;
- $C38$ is a by-pass capacitor.

Microcontroller Schematic (Part1)

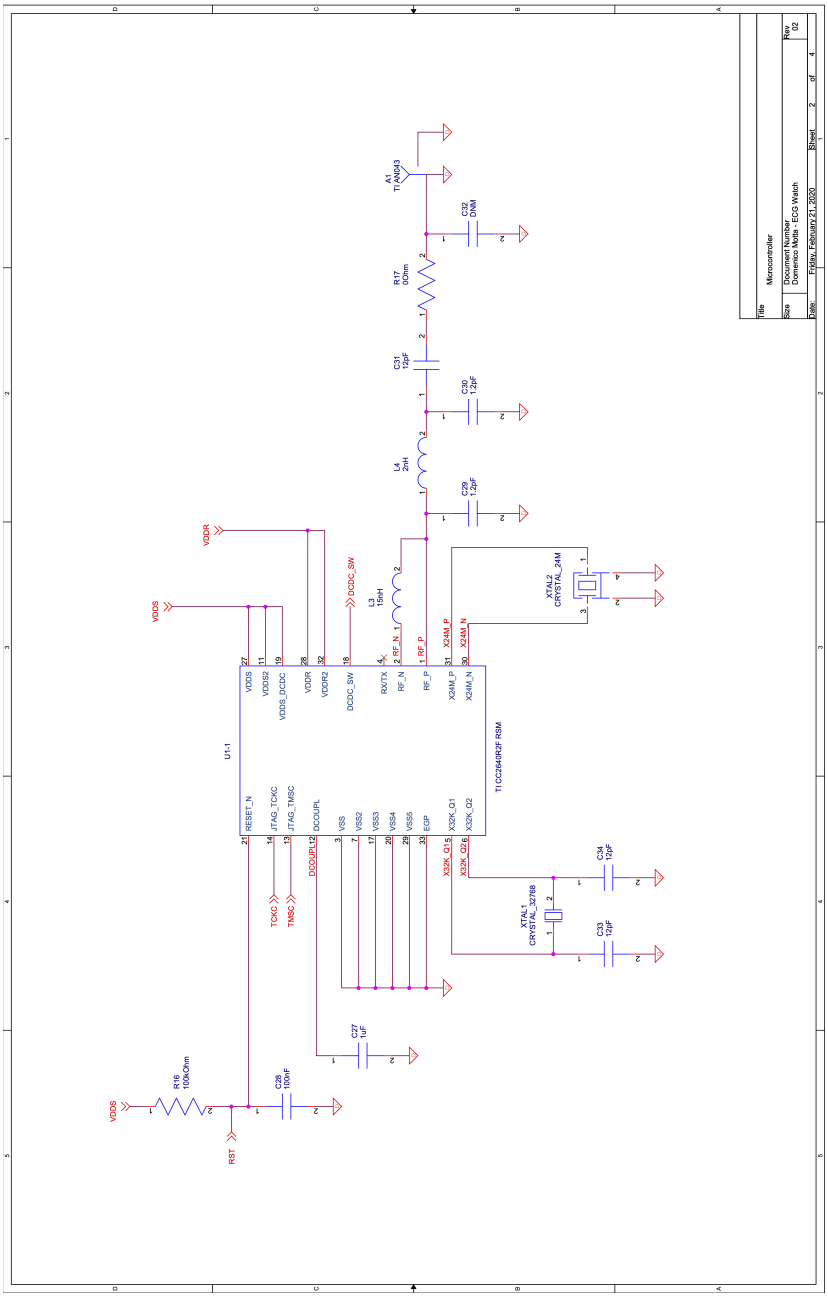


Figure 4.45. Microcontroller Schematic (Part1).

Simple explanation:
RF_N and RF_P pins are connected to the RF front-end circuit in order to send data from the internal Bluetooth module to the antenna.
Different capacitors are used to decouple the pins of the microcontroller.
Two external clock sources are used: in detail, the 24MHz crystal, required as the frequency reference for the radio, does not require decoupling capacitors.

Microcontroller DIOs and Flat Connector Schematic

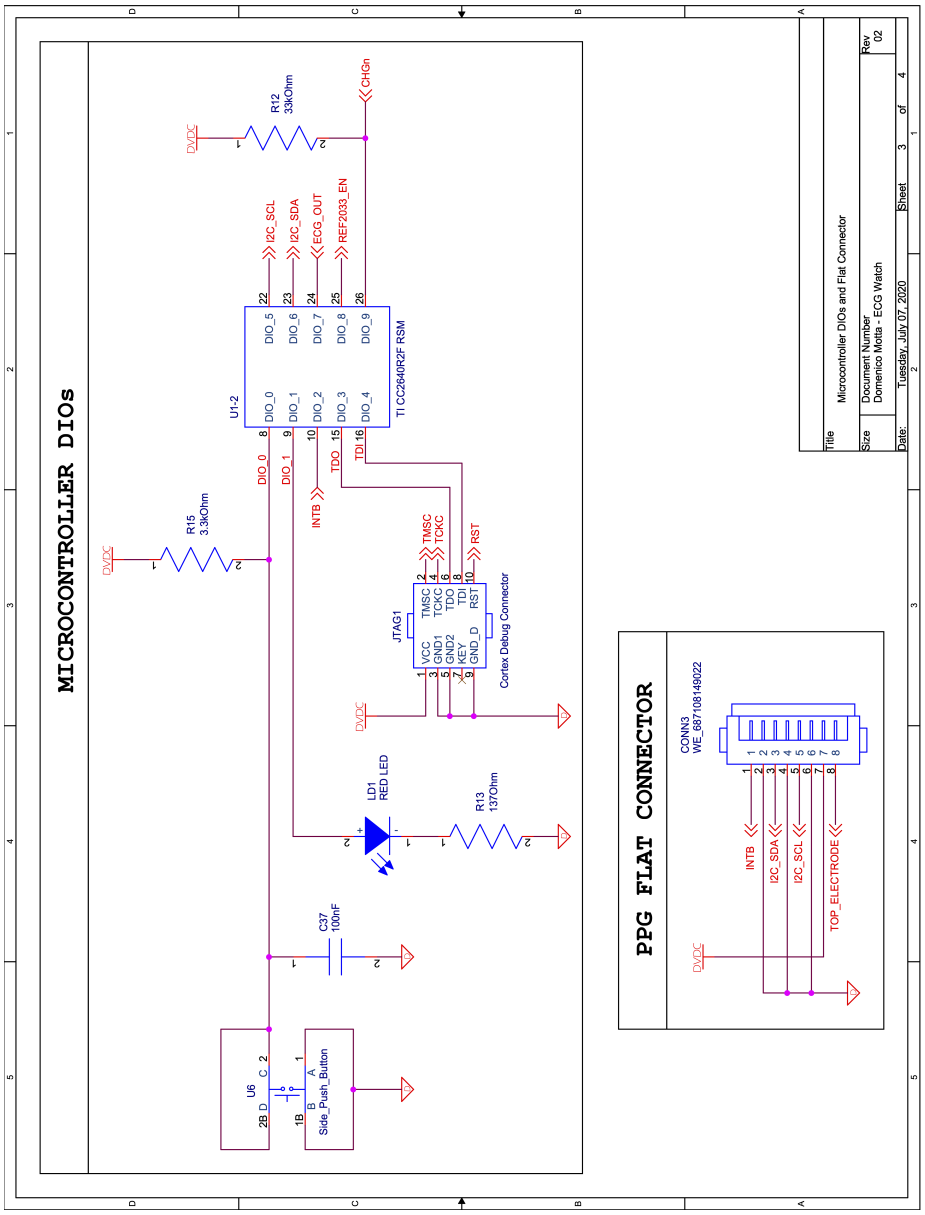


Figure 4.46. Microcontroller DIOs and Flat Connector Schematic.

Simple explanation:
JTAG_TMSC, JTAG_TCKC, DIO_3, DIO_4 and JTAG_RST are used to connect the microcontroller to the JTAG connector.
geq The pins DIO_0, DIO_1 are used to interact with the LED and with the push button; DIO_5 and DIO_6 are used for the I^2C communication; DIO_7 is the ADC input used to acquire the ECG signal while DIO_8 is the pin enable of the REF2033 component.
Flat connector is used to connect the main board to the external board of the MAXM86161 Sensor.

PPG Board

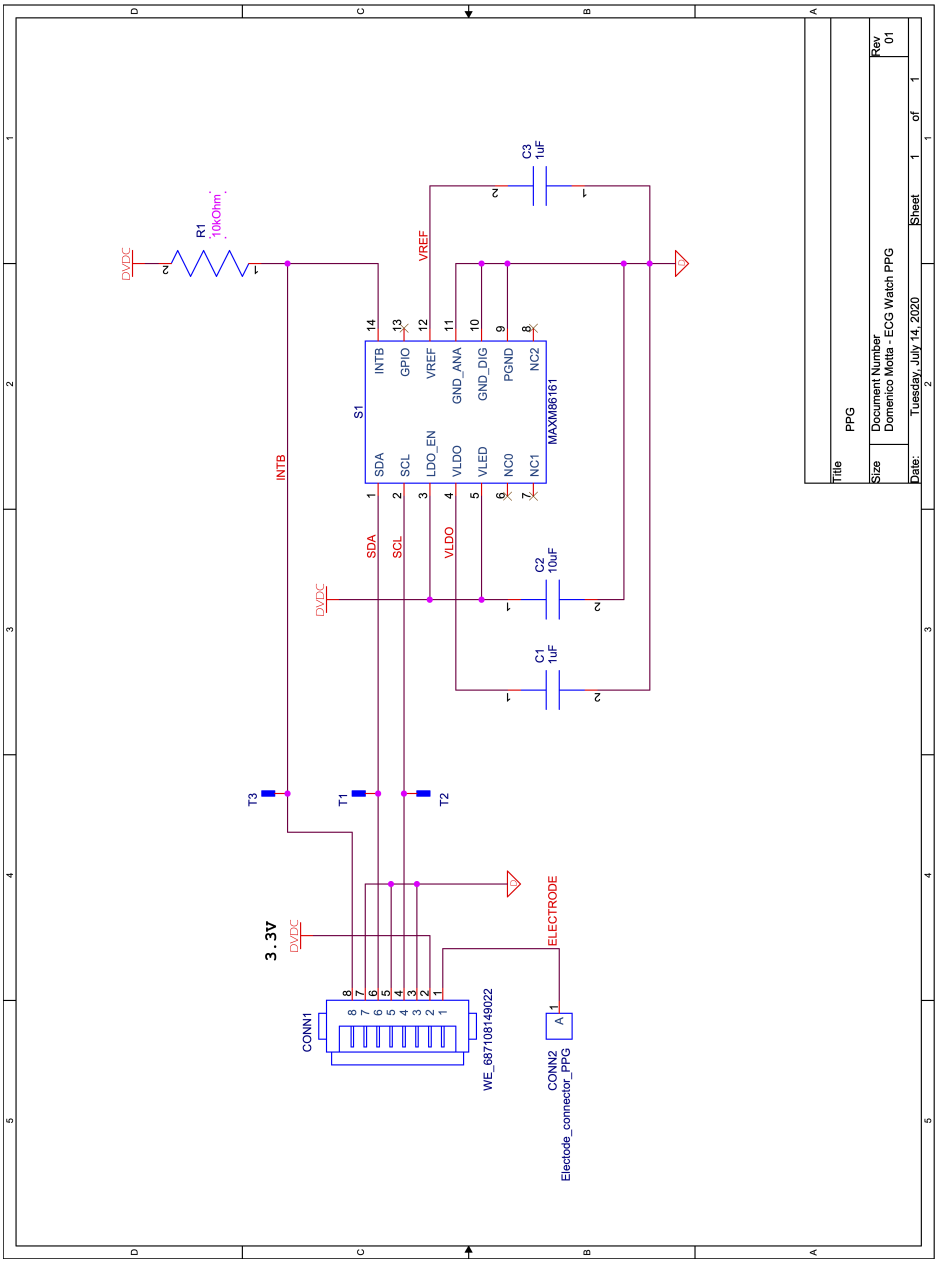
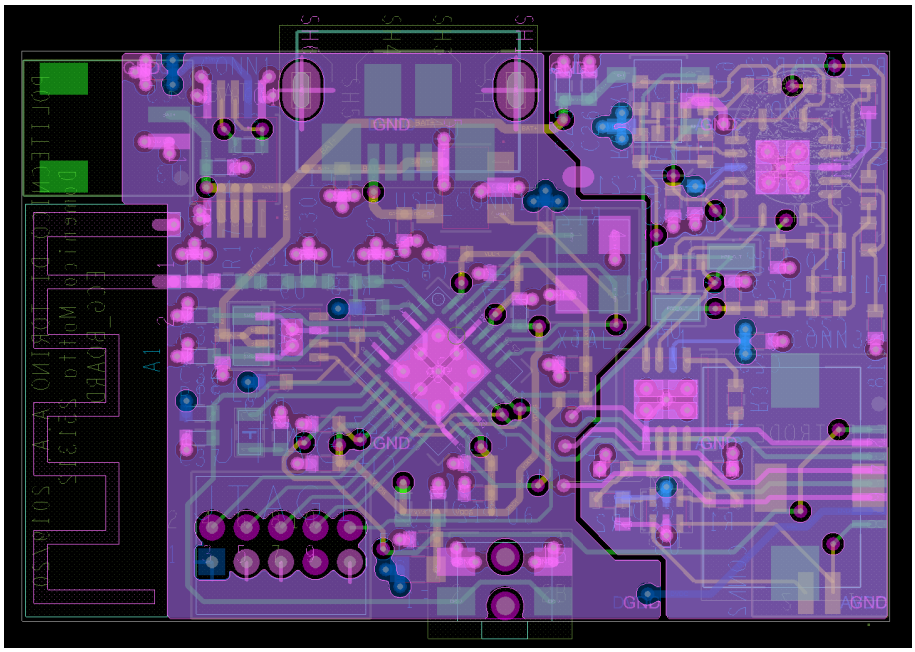
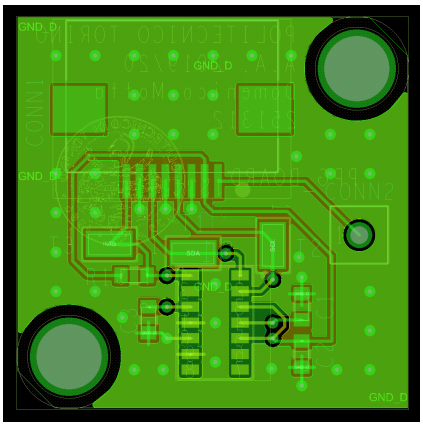


Figure 4.47. PPG Schematic.

4.3 Layout Explanation



(a) Main and ECG Board Layout.



(b) PPG Board Layout.

Figure 4.48. OrCAD Design.

After designing the schematic, the next phase is the PCB design. In order to do this, from Capture CIS tool has been exported the netlist and import it back into OrCAD PCB Designer tool. There are some rules that mus be respected to make this passage:

- The Antenna layout must be positioned near the edge and under it there must be no ground or power planes that shield the output signal.
- Antenna and its matching circuit must be positioned as near as possible to the microcontroller and in a straigth line design to avoid impedance mismatching.
- Decoupling capacitors and by-pass capacitors must be positioned near the related pin component.
- Analog and Digital Ground plans must be separated from each other apart in a very small bridge Figure 4.49.

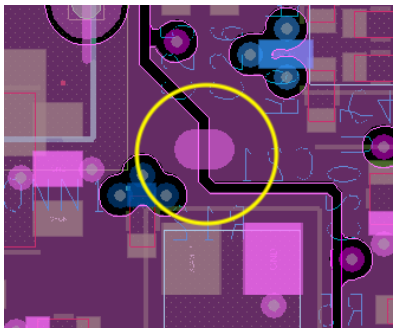


Figure 4.49. Small bridge.

- Analog and Digital Power Supply plans must be separated from each other Figure 4.3.
- Button and Connectors must be positioned near the edge.
- Ground loops must be avoided.
- Ground island not connected must be avoided.
- Component must be positioned with a logic sense and not random, so for example ECG front-end components have been placed in a specific area of the PCB.

- Component that are connected to Ground must ave a 3-point connection to guarantee a good current flow Figure 4.50.

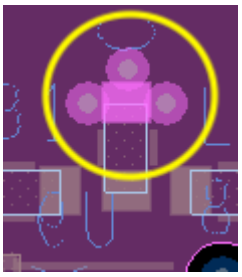
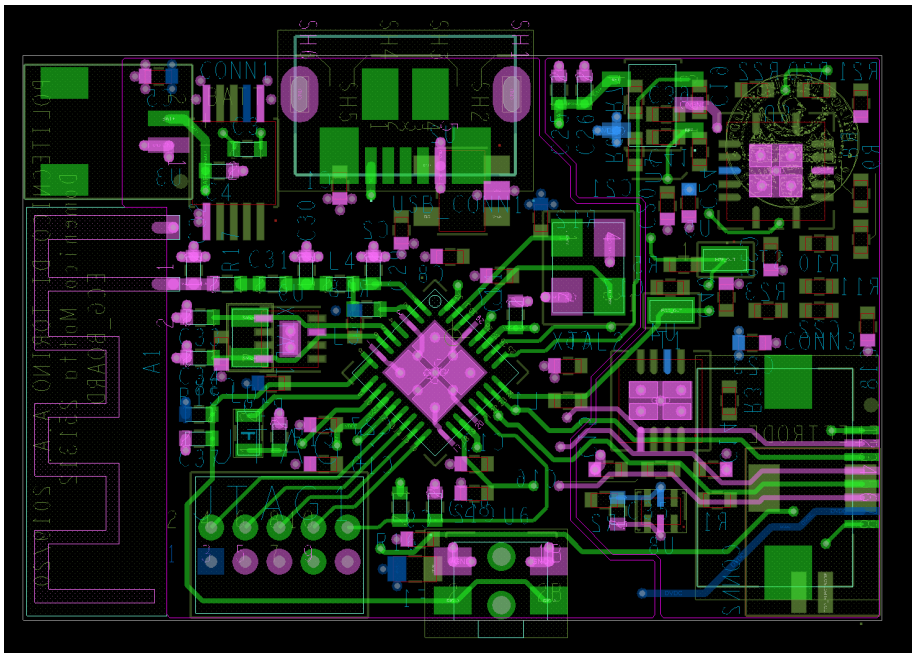


Figure 4.50. Three point connection.

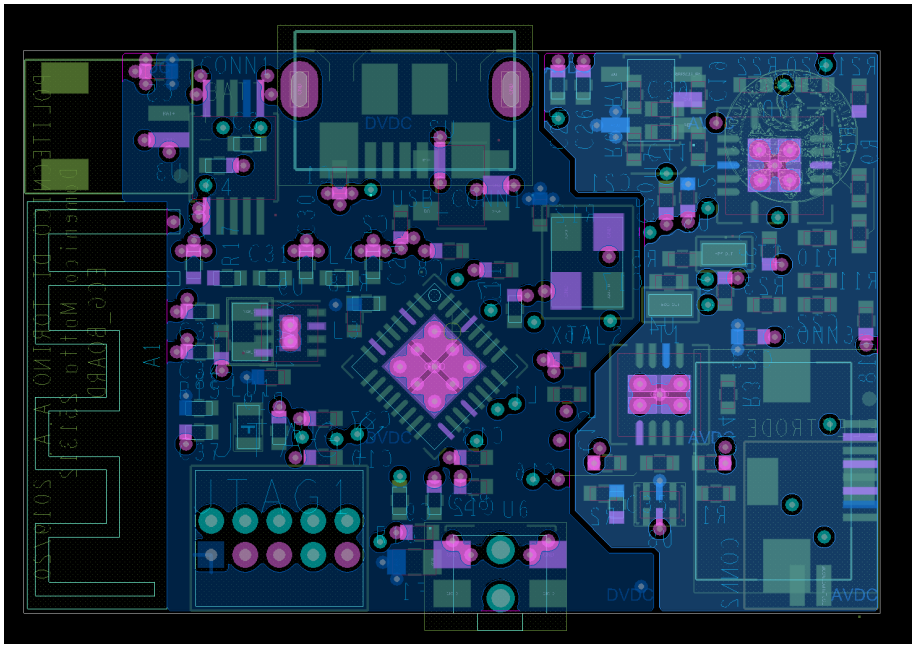
- Avoid any DRC Error that can cause electrical problems.

In this case the main PCB has a four layers design, while PPG board has only two layers. Next phase is the generation of Gerber files used by the manufacturer to print and build physical PCBs.

Below are reported some Gerber files output:

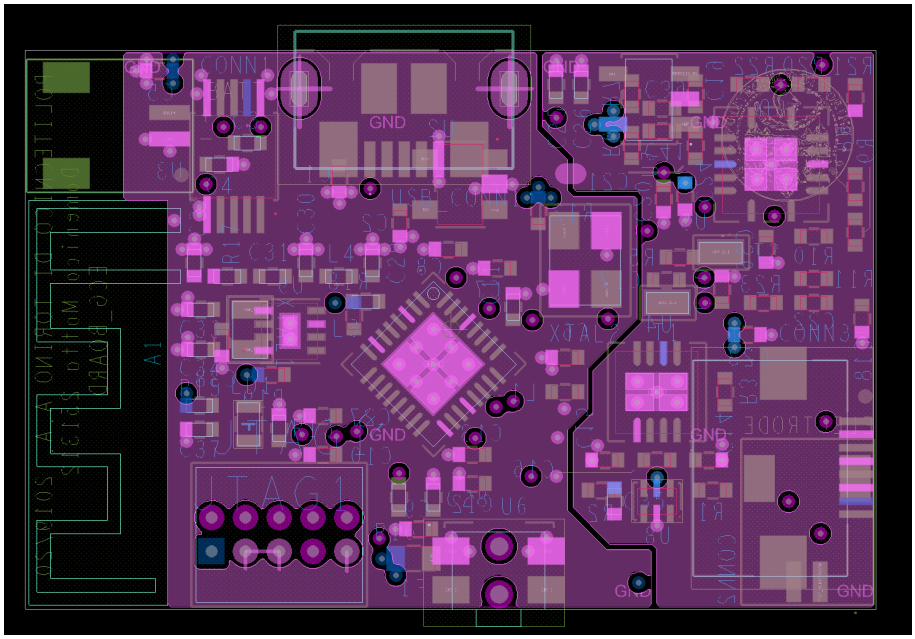


(a) ECG Gerber Top.

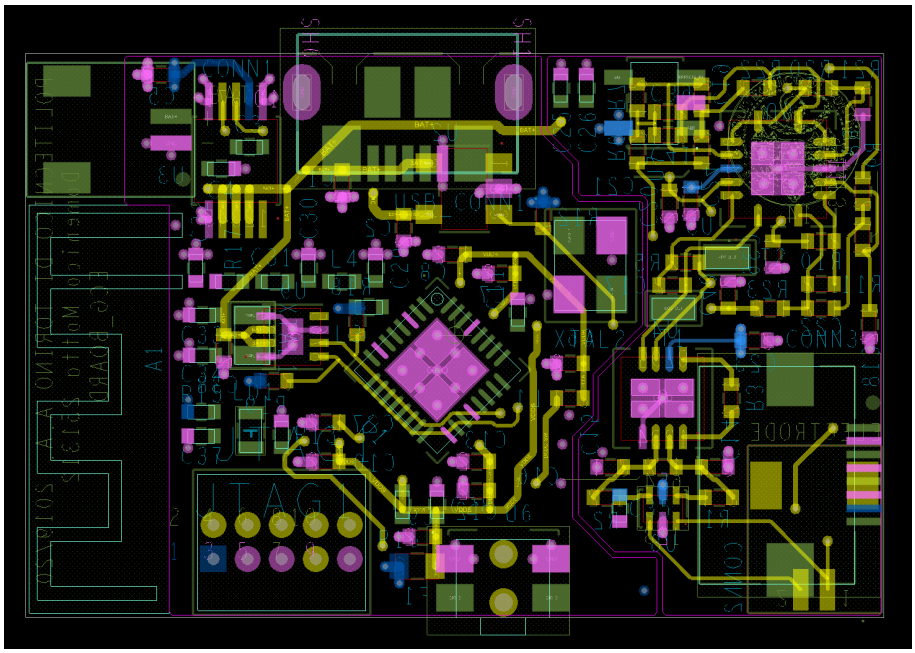


(b) ECG Gerber DVDC and AVDC.

Figure 4.51. Main and ECG Board Gerber Files.

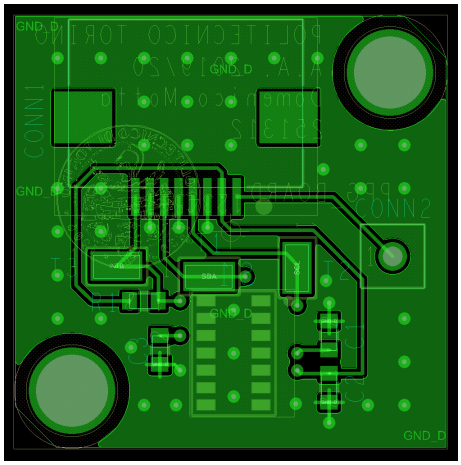


(a) ECG Gerber DGND and AGND.

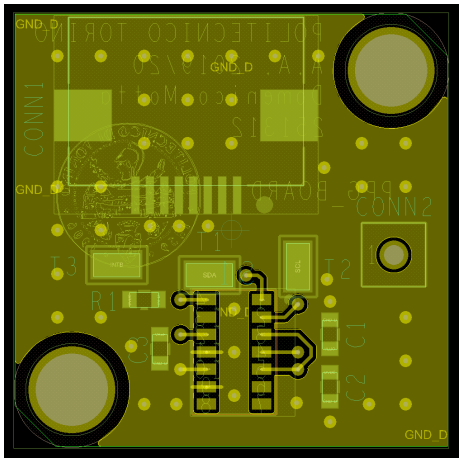


(b) ECG Gerber Bottom.

Figure 4.52. Other Main and ECG Board Gerber Files.



(a) PPG Gerber Top.



(b) PPG Gerber Bottom.

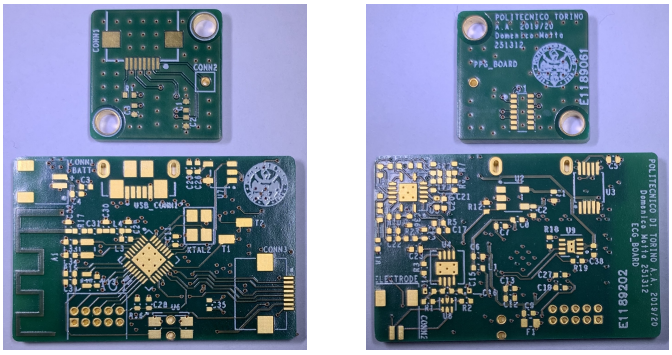
Figure 4.53. PPG Board Gerber Files.

4.4 Bill of Material

Item	Quantità	Descrizione	Codice Produttore
1	25	Connettori 2 Posizioni Basetta 0,039" (1,00m)	SM02B-SRSS-TB(LF)(SN)
2	2	Connettori FFC, FPC 8 posizioni Contatti, parte inferiore 0,020" (0,50mm)	687108149022
3	6	Condensatori ceramici 1µF ±10% 16V X5R 0402	EMK105BJ105KV-F
4	1	Condensatori ceramici 0,33µF ±10% 10V X7S 0402	C1005X7S1A334K050BC
5	5	Condensatori ceramici 10µF ±20% 10V X5R 0402	0402ZD106MAT2A
6	25	Condensatori ceramici 0,1µF ±20% 25V X5R 0402	885012105018
7	2	Condensatori ceramici 33pF ±5% 25V C0G, NP0 0402	C0402T330J3GACTU
8	1	Condensatori ceramici 330pF ±5% 50V C0G, NP0 0402	GCM1555C1H331JA16D
9	1	Condensatori ceramici 4,7µF ±10% 10V JB 0402	C1005JB1A475K050BC
10	1	Condensatori ceramici 0,047µF ±5% 25V X7R 0402	GRM155R71E473JA88D
11	4	Condensatori ceramici 0,15µF ±10% 10V X7R 0402	C1005X7R1A154K050BB
12	1	Condensatori ceramici 680pF ±5% 100V C0G, NP0 0402	CGA2B1C0G2A681J050BE
13	2	Condensatori ceramici 1,2pF ±0,1pF 50V C0G, NP0 0402	81-GJM1555C1H1R2BB01
14	3	Condensatori ceramici 12pF ±5% 50V C0G, NP0 0402	C0402C120J5GACTU
15	1	FLAT 8 Position FFC, FPC Cable 0,020" (0,50mm) 1,180" (29,97mm)	0152660073
16	1	Condensatori ceramici 10000pF ±5% 50V X7R 0402	GRM155R71H103JA88D
17	1	1,5 kOhms @ 100MHz 1 Linea del segnale Ferrite Bead 0603 500mA 500mOhm	BLM18HE152SN1D
18	1	Connettori 10 Posizioni Basetta, tagliabile 0,050" (1,27mm) Foro passante Oro	FTSH-105-01-L-D-K
19	1	Indicazione LED - Discreta Rosso 631nm 2V 0603	LTST-C193KRKT-5A
20	1	Induttore Multistrato Non schermato 10nH 500mA 260mOhm max 0402	LQG15HS10NJ02D
21	1	Induttore Multistrato Non schermato 15nH 450mA 320mOhm max 0402	LQG15HS15NJ02D
22	1	Induttore Multistrato Non schermato 2nH 900mA 100mOhm max 0402	LQG15HS2N0S02D
23	3	Resistori su chip A film sottile 100 kOhms ±0,5% 0,063W, 1/16W 0402	RR0510P-104-D
24	1	Resistori su chip A film sottile 24,9 kOhms ±0,1% 0,063W, 1/16W 0402	ERA-3AEB2492V
25	1	Resistori su chip A film sottile 68 kOhms ±0,5% 0,063W, 1/16W 0402	RR0510P-683-D
26	2	Resistori su chip A film sottile 21,3 kOhms ±1% 0,063W, 1/16W 0402	RN731ETTP2132F25
27	1	Resistori su chip A film spesso 243 kOhms ±1% 0,1W, 1/10W 0402	ERJ-2RKF2433X
28	1	Resistori su chip A film spesso 330 kOhms ±0,5% 0,063W, 1/16W 0402	ERJ2RKD3303X
29	1	Resistori su chip A film sottile 1 kOhms ±0,5% 0,063W, 1/16W 0402	ERA-2AED102X
30	1	Resistori su chip A film sottile 150 kOhms ±0,1% 0,063W, 1/16W 0402	CPF0402B150KE1
31	1	Resistori su chip A film sottile 33 kOhms ±0,1% 0,063W, 1/16W 0402	ERA-2AEB333X
32	1	Resistori su chip A film spesso 137 Ohms ±1% 0,063W, 1/16W 0402	RC0402FR-07137RL
33	1	Resistori su chip A film sottile 3,3 kOhms ±0,1% 0,063W, 1/16W	ERA-2AEB332X
34	1	Resistori su chip A film spesso 0 Ohms Ponticello 0,063W, 1/16W 0402	RMCF0402ZT0R00
35	4	Resistori su chip A film sottile 10 kOhms ±0,5% 0,063W, 1/16W 0402	ERA-2AED103X
36	1	Resistori su chip A film sottile 1,1 kOhms ±0,1% 0,063W, 1/16W 0402	ERA-2ARB112X
37	1	Resistori su chip A film sottile 10,7 kOhms ±0,1% 0,063W, 1/16W 0402	CPF0402B10K7E1
38	1	Resistori su chip A film sottile 107 kOhms ±0,1% 0,1W, 1/10W 0402	RP73PF1E107KBTD
39	5	Test Point TE RCU-0C	RCU-0C
40	1	USB CONNECTOR MOLEX 1051640001	1051640001
41	1	IC RF TXRX+MCU CC2640R2FRSMR BLUETOOTH 32VFQFN	CC2640R2FRSMR
42	1	MAXIM MAX1555	MAX1555EZK+T
43	1	MAXIM MAX1759	MAX1759EUB+
44	1	TI INA333	INA333AIDRGR
45	1	TI_OPA4330	OPA4330AIRGYT
46	1	Interruttore tattile SPST-NO Ad azionamento laterale Montaggio superficiale	EVQ-P7C01P
47	1	TI_REF2033	REF2033AIDDCR
48	1	17V Clamp 5 A (8/20µs) Ipp Tvs Diode A montaggio superficiale 6-µQFN	DVIULC6-2M6
49	1	Battery Monitor batteria IC Ioni di litio 8-TDFN-EP (2x2)	MAX17048G+T10
50	1	CRYSTAL 32.7680KHZ 12.5PF SMD	SC20S-12.5PF20PPM
51	1	CRYSTAL 24.0000MHZ 9PF SMD	TSX-3225 24.0000MF15X-AC3
52	1	OPTICAL BIO SENSOR MODULE MAXM86161	MAXM86161EFD+

Figure 4.54. BOM.

4.5 Mounting Process



(a) Top view. (b) Bottom view.

Figure 4.55. PCBs Received from the Manufacturer.

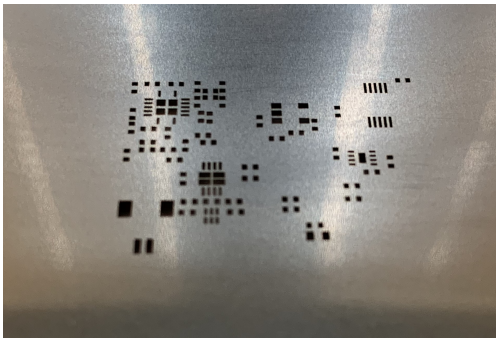
The mounting process has been performed after the manufacturer produce PCBs and Stencils starting from Gerber Files and has been divided in these steps in order to have a final result shown in Figure 4.56:

- **Soldering Paste Spreading;**
- **Component Placing;**
- **Reflow Oven Soldering.**

Figure 4.56. Final Result.

4.5.1 Soldering Paste Spreading

This step involves inserting the stencil, a laser cut foil, into the stencil mate and stretching it as in Figure 4.5.1.



(a) Stencil.

(b) Stencil Mate with stretched Stencil.

Figure 4.57. Stencil Preparation.

After putting stencil and pcb in contact, take soldering paste (Figure 4.58) and try to spatulate it in holes that match to the padstack of every SMD component, so as to release a small layer of pasta only where it is needed.

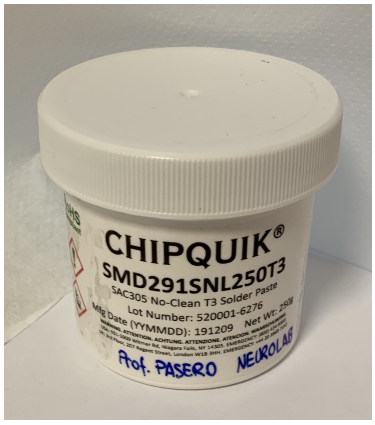


Figure 4.58. Soldering Paste.

4.5.2 Component Placing

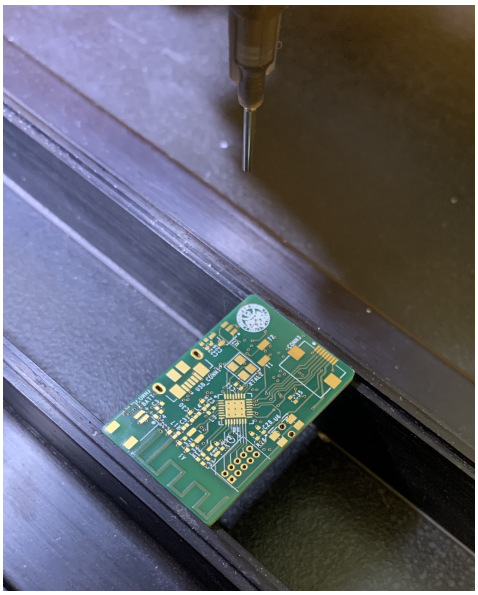


Figure 4.59. PCB positioned in the PickPlace Machine and the needle over it.

This phase involves positioning the SMD components on the PCB. Since they are very small and, since much precision is needed, a machine called PickPlace Machine is used Figure 4.60. It is able to collect the components through a needle, as it approaches the component the vacuum is created and the component can then be positioned on the PCB; in contact with PCB's surface the vacuum turn off. This machine has four degrees of freedom (x, y, z axis and rotation) so as to allow the correct positioning of the component.

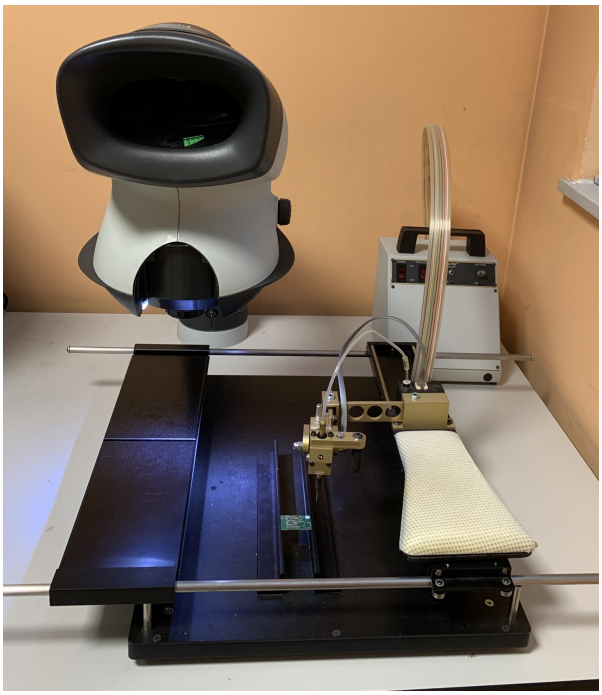


Figure 4.60. PickPlace Machine.

These steps must be repeated for all available components. At the end the complete PCB can be brought to the soldering phase.

4.5.3 Reflow Oven Soldering

This phase allows to solder the components, positioned on the paste, to the PCB. This is done by placing the circuit in the Reflow Oven Figure 4.61.

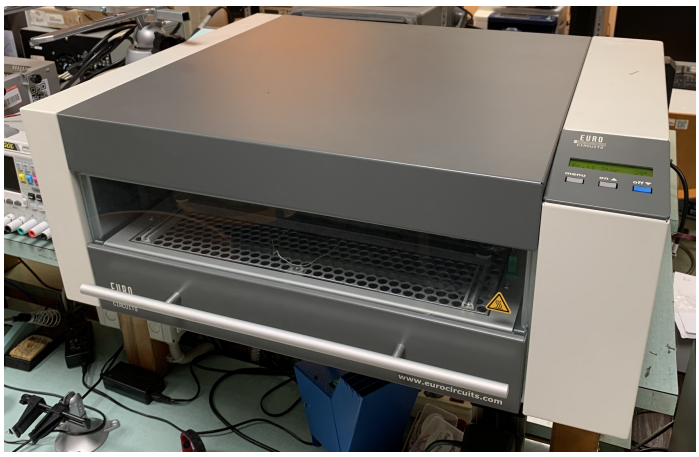


Figure 4.61. Reflow Oven.

This oven has the ability to manage temperature profiles so that it can do its job without damaging the components.The profile shown in the Figure 4.62 has been chosen and below you can see the progress of the oven.

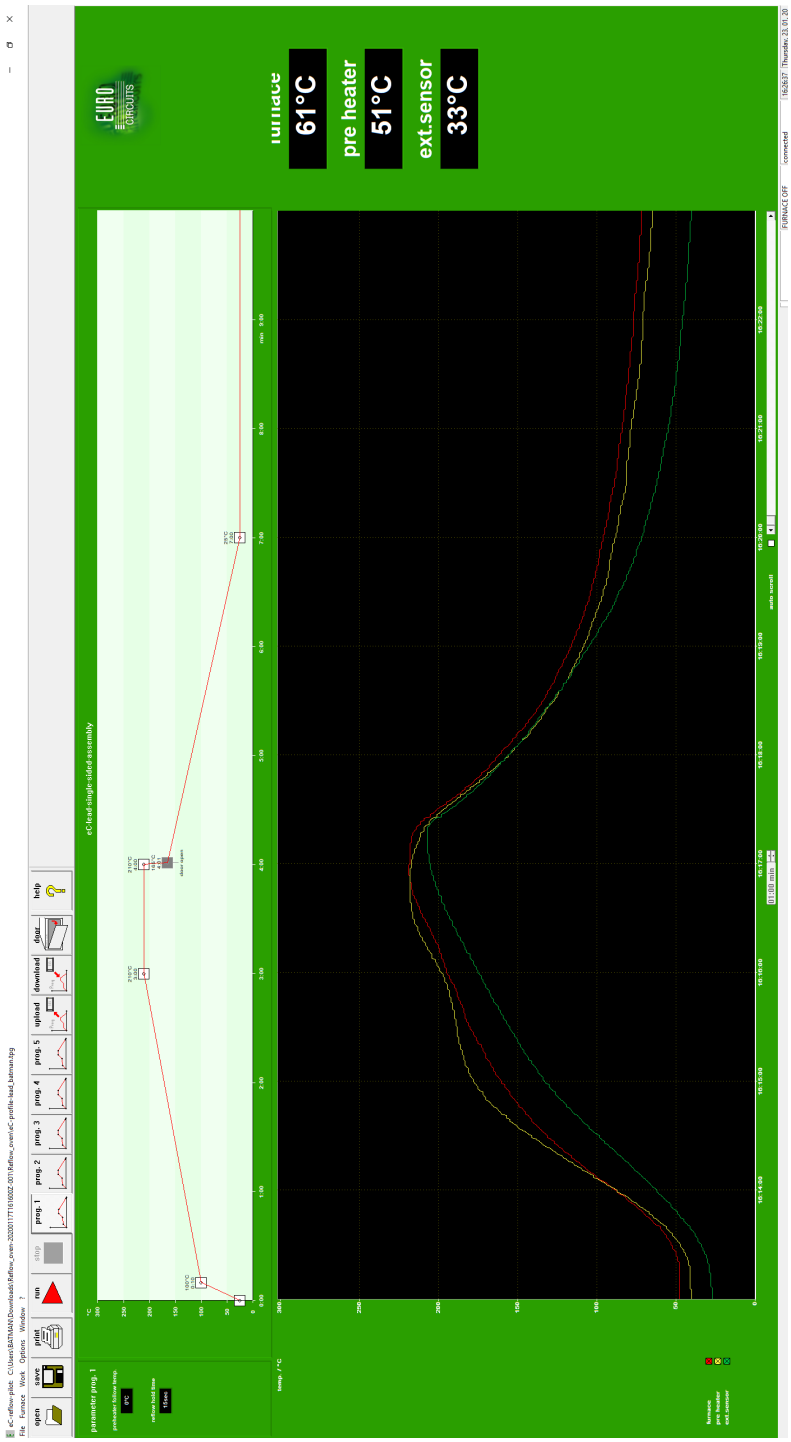


Figure 4.62. Temperature Profile.

Chapter 5

Firmware

In this chapter the Firmware part of the project is described. As previously said, a Texas Instrument microcontroller has been used, it has the possibility of working with a real-time operating system (RTOS). This is an operating system which gives the possibility to run multiple threads at the same time so that, the execution speed of the code can be increased. To decide the thread to be run, by default, it has a scheduling called Preemptive scheduling. This allows to run a thread until, for example, another thread with a higher priority requires processor use, or run until it end like an interrupt service routine. Two Texas Instrument tools have been used during programming:

- **Code Composer Studio**
- **Sensor Controller Studio**

This because the microcontroller has inside a co-processor which can be used to facilitate peripheral management.

Figure 5.1. Code Composer Studio Logo.

5.1 Sensor Controller Studio

As mentioned before, this tool was used in order to program the co-processor called Sensor Controller implemented so as facilitate the management of peripherals such as data exchange between the microcontroller and MAX17048, for reading the voltage value and the battery state of charge. Another peripheral managed by the Sensor Controller is the ADC used to convert from analog to digital data coming from the output of the ECG front-end. The

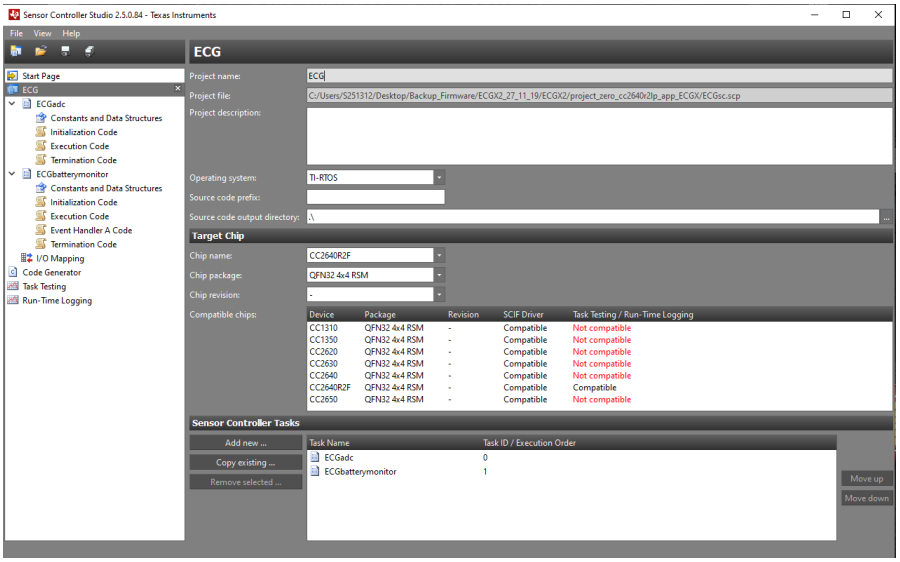


Figure 5.2. Screenshot of SCS Tool main page project.

project has been organized in two tasks: one for the ECG ADC and one for ECG battery monitor. Each task can manage a certain number of resources such as ADC, I^2C Communication, Interrupt, Digital Output Pin System CPU Alert and Math and Logic unit. Each task consists of three/four main functions depending on whether or not interrupts are used. These functions are:

- *Initialization Code*: ran only once at the start;
- *Execution Code*: repeated on request;
- *Event Handler a Code*: used as Interrupt service routine;
- *Termination Code*: used in some case to shut down a device or reset characteristics and variables.

5.1.1 Sensor Controller Tasks

In this project for *ECGadc* task are used the following resources:

- **General-Purpose I/O:** used in order to manage Analog pin for ADC and Digital Pins for LED Red and REF2033 Enable;
- **Peripherals:** used for programming ADC Converter and internal Timer;
- **System CPU Communication:** used for generate alert interrupt from task code to Code Composer Code;
- **Other Utilities:** such as Math and Logic Operators.

Regarding *ECGbatterymonitor* task are used the following resources:

- **General-Purpose I/O:** used in order to manage Digital Pins for LED Red;
- **Serial Interfaces:** used for programming I^2C Master interface;
- **System CPU Communication:** used to generate alert interrupt from task code to Code Composer Code;
- **Task Event Handling:** used for handling Timer 1 Event.

Pin mapping and description of each task are shown below.

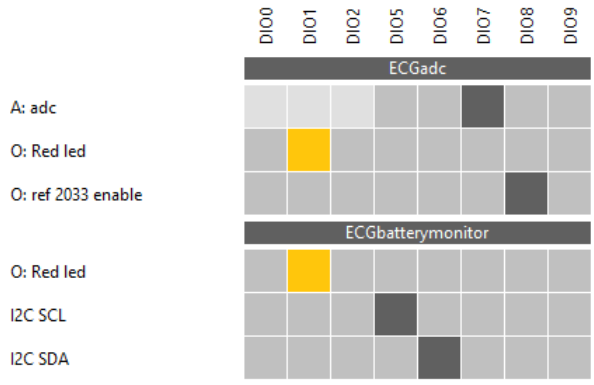


Figure 5.3. Pin Mapping.

ECGadc

This task is used to program the ADC. Below are reported all the scripts.

Constants and Data Structures

Figure 5.4. Constants and Data Structures.

Execution Code

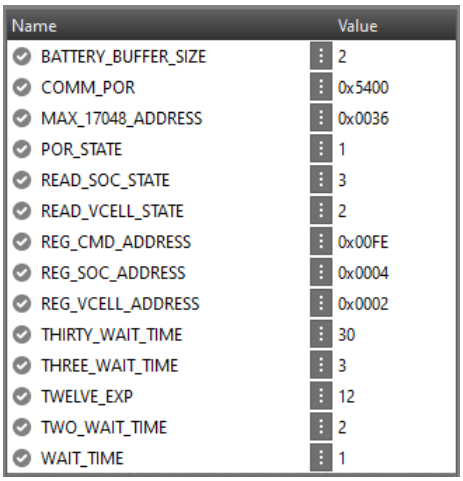
```
// Set ON the Green Led
gpioSetOutput(AUXIO_0_G_LED);

// Ref 2033
gpioSetOutput(AUXIO_0_REF2033_EN);

// Select ADC input
adcSelectGpioInput(AUXIO_A_ADC_IN);

// Enable the ADC
adcEnableSync(ADC_REF_FIXED, ADC_SAMPLE_TIME_2P7_US, ADC_TRIGGER_AUX_TIMER0);

// Start ADC trigger timer at 2ms (500Hz) as 24MHz/(24000*2^1)
timer0Start(TIMER0_MODE_PERIODICAL, 24000, 1);
//timer0Start(TIMER0_MODE_PERIODICAL, 60000, 2);

U16 n;
n = 0;
U16 i;
i = 0;

state.enabled = 1;

// Loop until the application sets the exit flag
while ( i<input.length ){
    i = i+1;
    //utilIncrAndWrap( i, ITER; i );

    n = state.head;
    adcReadFifo( output.Data[n] );
    utilIncrAndWrap( n, BUFFER_SIZE; state.head );

    if( state.head == 0 ){
        fwGenQuickAlertInterrupt();
        // Turn ON the Green Led
        gpioSetOutput(AUXIO_0_G_LED);
    }

    if( state.head == HALF_BUFFER ){
```

```
40          fwGenQuickAlertInterrupt();
41          // Turn OFF the Green Led
42          gpioClearOutput(AUXIO_O_G_LED);
43      }
44
45  }
46
47  // Stop the ADC trigger and flush the ADC FIFO
48  timer0Stop();
49  adcFlushFifo();
50
51  // Disable the ADC
52  adcDisable();
53
54  state.enabled = 0;
55
56  // Turn OFF the Green Led
57  gpioClearOutput(AUXIO_O_G_LED);
```

ECGbatterymonitor

This task is used to communicate with I^2C every 30s with MAX17048. Below are reported all the scripts.

Constants and Data Structures



Name	Value
BATTERY_BUFFER_SIZE	2
COMM_POR	0x5400
MAX_17048_ADDRESS	0x0036
POR_STATE	1
READ_SOC_STATE	3
READ_VCELL_STATE	2
REG_CMD_ADDRESS	0x00FE
REG_SOC_ADDRESS	0x0004
REG_VCELL_ADDRESS	0x0002
THIRTY_WAIT_TIME	30
THREE_WAIT_TIME	3
TWELVE_EXP	12
TWO_WAIT_TIME	2
WAIT_TIME	1

Figure 5.5. Constants and Data Structures.

Initialization Code

```
//-----------------------------------------------------------------------
// Inizialization of the timer in order to wait "WAIT_TIME" [s] interval
//-----------------------------------------------------------------------
evhSetupTimer1Trigger(0, WAIT_TIME, TWELVE_EXP);
state.timer = POR_STATE;
state.test = 1;
```

Event Handler A Code

```
//-------------------------------------------------------
//Apply POR (Power-On-Reset); battery debounce can cause an error in the first SOC and VCELL
     estimation
//-----------------------------------------------
if(state.timer == POR_STATE) {
    state.test = 2;
    //----------------------------
    // POR
    //----------------------------
    i2cStart();
    i2cTx((MAX_17048_ADDRESS<<1) | I2C_OP_WRITE);
    i2cTx(REG_CMD_ADDRESS);
    i2cTx(COMM_POR);
    // Inizialization of the timer in order to wait first VCELL and SOC reading "WAIT_TIME"
     [s] interval
    evhSetupTimer1Trigger(0, WAIT_TIME, TWELVE_EXP);
```

```
15      gpioSetOutput(AUXIO_O_R_LED);
16      i2cStop();
17      state.timer = READ_VCELL_STATE;
18  //-------------------------------------------------
19  // Configure and start the VCELL and SOC measurement
20  //-------------------------------------------------
21  } else {
22      //----------------------------
23      //VCELL reading
24      //----------------------------
25      if (state.timer == READ_VCELL_STATE) {
26          i2cStart();
27          i2cTx((MAX_17048_ADDRESS<<1) | I2C_OP_WRITE);
28          i2cTx(REG_VCELL_ADDRESS);
29          i2cRepeatedStart();
30          i2cTx((MAX_17048_ADDRESS<<1) | I2C_OP_READ);
31
32          U16 vcellMSB;
33          U16 vcellLSB;
34
35          //Read VCELL value
36          i2cRxAck(vcellMSB);
37          i2cRxAck(vcellLSB);
38          i2cStop();
39
40          U16 vcell_value = ((vcellMSB<<8) | vcellLSB);
41
42          if(vcell_value > 2000) {
43              state.test = 9;
44              // Put values in output vector and notify the application with an alert
45              output.Battery[0] = vcell_value;
46
47              // Inizialization of the timer in order to wait next SOC reading "WAIT_TIME" [s]
     interval
48              evhSetupTimer1Trigger(0, WAIT_TIME, TWELVE_EXP);
49              gpioSetOutput(AUXIO_O_R_LED);
50              state.timer = READ_SOC_STATE;
51          } else {
52              state.test = 109;
53              // Inizialization of the timer in order to wait next SOC reading "WAIT_TIME" [s]
     interval
54              evhSetupTimer1Trigger(0, WAIT_TIME, TWELVE_EXP);
55              state.timer = READ_VCELL_STATE;
56          }
57      //----------------------------
58      //SOC reading
59      //----------------------------
60      } else {
61          i2cStart();
62          i2cTx((MAX_17048_ADDRESS<<1) | I2C_OP_WRITE);
63          i2cTx(REG_SOC_ADDRESS);
64          i2cRepeatedStart();
65          i2cTx((MAX_17048_ADDRESS<<1) | I2C_OP_READ);
66
67          U16 socMSB;
68          U16 socLSB;
69
70          //Read SOC value
```

```
71          i2cRxAck(socMSB);
72          i2cRxAck(socLSB);
73          i2cStop();
74
75          U16 soc_value = ((socMSB<<8) | socLSB);
76
77          // Put values in output vector and notify the application with an alert
78          output.Battery[1] = soc_value;
79          // Alert Interrupt used to inform the Code Composer that values of VCELL and SOC are
        present in output
80          fwGenAlertInterrupt();
81          state.test = 13;
82
83          // Inizialization of the timer in order to wait next VCELL reading "THIRTY_WAIT_TIME
        " [s] interval
84          evhSetupTimer1Trigger(0, THIRTY_WAIT_TIME, TWELVE_EXP);
85          gpioClearOutput(AUXIO_0_R_LED);
86          state.timer = READ_VCELL_STATE;
87          state.test = 14;
88      }
89  }
```

Termination Code

```
1  //-----------------------------------------------------------------------------------------
2  // If the System CPU application stops the task, cancel the potentially active event trigger
3  //-----------------------------------------------------------------------------------------
4  evhCancelTrigger(0);
5  gpioClearOutput(AUXIO_0_R_LED);
```

5.2 Code Composer Studio

In this section has been analyzed the part of the code written on Code Composer Studio. In order to use bluetooth features, the firmware project was not created from scratch due to high complexity of BLE stack, but has been started from a project called *ProjectZero* in which, Sensor Controller management and some parts related to communication with the MAXM86161 sensor were added.

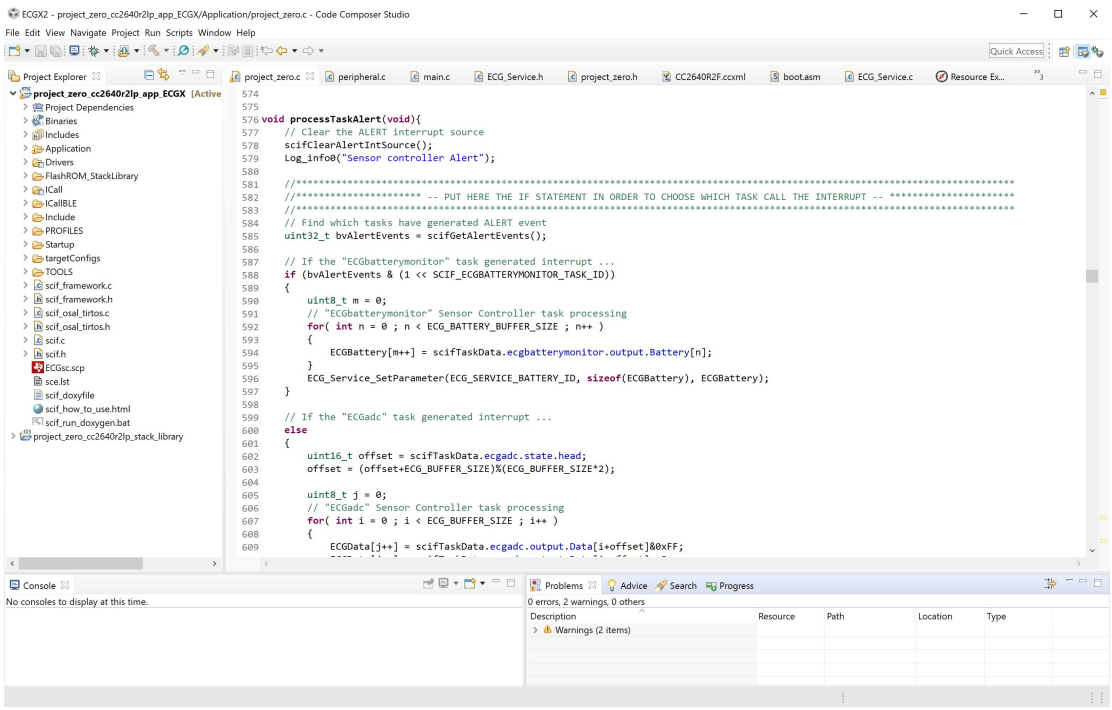


Figure 5.6. Screenshot of CCS Tool.

5.2.1 ProjectZero Main Procedures

ProjectZero_init()

This function is called before the task loop and contains all application initialization of the blutooth, hardware initialization and BLE profile/service initialization. "Board.h" is used to the GPIO initialization , where the PINs intended to be used with I^2C or simply as LED should be defined.

ProjectZero_taskFxn()

This function is the application task entry point. Inside is contained all the initialization functions call, it also contains the initialization functions for Sensor Controller and an infinite loop.

5.2.2 Sensor Controller Interface Functions

In order to take advantage of the functionality of the Sensor Controller, some directives must be respected. There are some functions that must be used for a correct communication between main core and this secondary core.

Initialization of SCIF Driver

The following function must be added in the application main function (ProjectZero_taskFxn()) and are used to initialize the sensor controller.

```
// Initialize the Sensor Controller
  scifOsalInit();
  scifOsalRegisterCtrlReadyCallback( scCtrlReadyCallback );
  scifOsalRegisterTaskAlertCallback( scTaskAlertCallback );
  scifInit( &scifDriverSetup );
```

scCtrlReadyCallback and *scTaskAlertCallback* are two callbacks that are used to manage data transfer between Sensor Controller processor and the main core.

Start of Sensor Controller Tasks

After the initialization there must be the Tasks start. It can be positioned or before the infinite loop so that the task always remains active or into the infinite loop so that it can be activated when needed.

```
//*****************************************************************
// Start the "ECGbatterymonitor" Sensor Controller task
// Every 30s the task read and put in output the SOC and VCELL value
// At the start the task make a Power-On Reset POR
scifStartTasksNbl(1 << SCIF_ECGBATTERYMONITOR_TASK_ID);
//*****************************************************************
```

For the "ECGbatterymonitor" Sensor Controller task since it has interrupt handling, can be left to run on its own or it can be trigger when has been wanted, this is done as follows:

```
// This code generates the event trigger for task ECGbatterymonitor
    scifSwTriggerEventHandlerCode();
```

The "ECGadc" Sensor Controller task on the contrary the start of the task start when it is necesssary by the function:

```
scifSwTriggerExecutionCodeNbl(1 << SCIF_ECGADC_TASK_ID);
```

Access to Sensor Controller Data Structure

Access data from Sensor Controller by the main core, can be done with a function called when Sensor Controller send an Alert. Inside can be read or write data structure while task is running. An example of usage is reported below.

```
void processTaskAlert(void){
    // Clear the ALERT interrupt source
    scifClearAlertIntSource();
    Log_info0("Sensor controller Alert");

    // Find which tasks have generated ALERT event
    uint32_t bvAlertEvents = scifGetAlertEvents();

    // If the "ECGbatterymonitor" task generated interrupt ...
    if (bvAlertEvents & (1 << SCIF_ECGBATTERYMONITOR_TASK_ID))
    {
        uint8_t m = 0;
        // "ECGbatterymonitor" Sensor Controller task processing
        for( int n = 0 ; n < ECG_BATTERY_BUFFER_SIZE ; n++ )
        {
            ECGBattery[m++] = scifTaskData.ecgbatterymonitor.output.Battery[n];
        }
        ECG_Service_SetParameter(ECG_SERVICE_BATTERY_ID, sizeof(ECGBattery), ECGBattery);
    }

    // If the "ECGadc" task generated interrupt ...
    else
    {
        uint16_t offset = scifTaskData.ecgadc.state.head;
        offset = (offset+ECG_BUFFER_SIZE)%(ECG_BUFFER_SIZE*2);

        uint8_t j = 0;
        // "ECGadc" Sensor Controller task processing
        for( int i = 0 ; i < ECG_BUFFER_SIZE ; i++ )
        {
            ECGData[j++] = scifTaskData.ecgadc.output.Data[i+offset]&0xFF;
            ECGData[j++] = scifTaskData.ecgadc.output.Data[i+offset]>>8;
        }
        ECG_Service_SetParameter(ECG_SERVICE_DATA_ID, sizeof(ECGData), ECGData);
    }

    // Acknowledge the ALERT event
    scifAckAlertEvents();
} // processTaskAlert
```

5.2.3 Bluetooth Services

In order to make the application run with Bluetooth, a new ad-hoc profile has been generated. Texas Instruments created an online tool that is easy to use and help the creation of the .c and .h files for the BLE profile and its characteristics Figure 5.7.

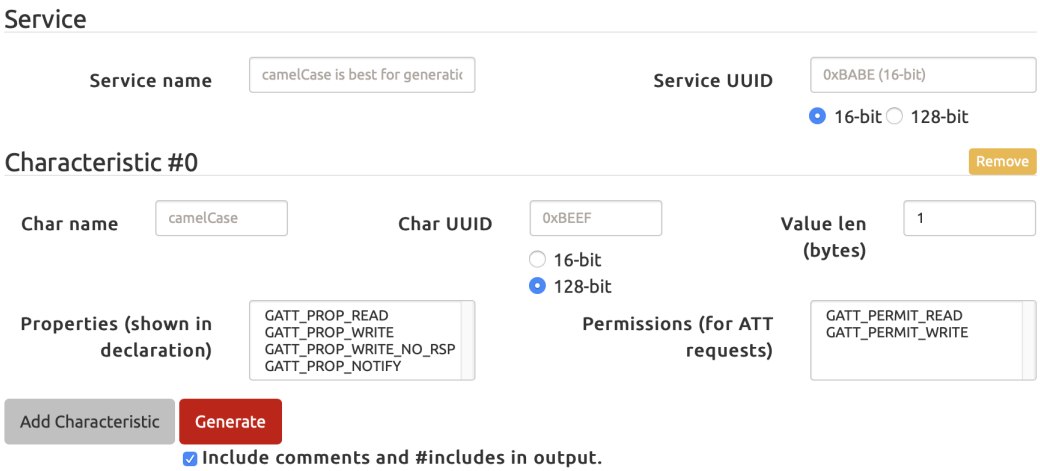


Figure 5.7. BLE Service TI Tool.

The ECG Service (*UUID: 0xBABE*) contain 3 characteristics:

- **ECG Data Service:** used to transfer ECG data packets from microcontroller to tablet app;
- **ECG Start Service:** used for start the ECG acquisition sending a value from tablet app to microcontroller;
- **ECG Battery Service:** used to transfer Battery data from microcontroller to tablet app.

Characteristics definition

```
    //  Characteristic defines
#define ECG_SERVICE_DATA_ID         0
#define ECG_SERVICE_DATA_UUID       0xECDA
#define ECG_SERVICE_DATA_LEN        200

    //  Characteristic defines
#define ECG_SERVICE_ECGSTART_ID     1
#define ECG_SERVICE_ECGSTART_UUID   0xECEC
#define ECG_SERVICE_ECGSTART_LEN    1

    //  Characteristic defines
#define ECG_SERVICE_BATTERY_ID      2
#define ECG_SERVICE_BATTERY_UUID    0xECBA
#define ECG_SERVICE_BATTERY_LEN     4
```

5.2.4 PPG Interrupt and I²C Management

PPG Sensor has not been managed via Sensor Controller, but is managed directly by the main core. This implies the management of two pins o for data transmission via I²C and one as interrupt pin. The interrupt pin is used to communicate the status of the sensor in two phases: one is for activating the acquisition via proximity sensor, the other one for communicating the micro when the internal FIFO inside PPG sensor has at least one value. Below has been reported some parts of the code.

Interrupt Management

Interrupt Pin Initialization

```
#define PIN_INTB IOID_2

/* Pin driver handles */
static PIN_Handle intBPinHandle;

/* Global memory storage for a PIN_Config table */
static PIN_State intBPinState;
/*
 * Application INTB pin configuration table:
 *   - INTB interrupts are configured to trigger on falling edge.
 */
PIN_Config intBPinTable[] = {
    PIN_INTB | PIN_INPUT_EN | PIN_PULLUP | PIN_IRQ_NEGEDGE,
    PIN_TERMINATE
};

...

/* Open INTB pins */
intBPinHandle = PIN_open(&intBPinState, intBPinTable);
if(!intBPinHandle) {
  Log_info0("Error initializing INTB pins");
  while(1);
}
```

Interrupt Callback Function

```
\\ Function Initialization
void intBCallbackFxn(PIN_Handle handle, PIN_Id pinId);

...

/* Setup callback for intB pins */
if (PIN_registerIntCb(intBPinHandle, &intBCallbackFxn) != 0) {
  Log_info0("Error registering intB callback function");
  while(1);
}

...

```

```
/*
 *  ======== intBCallbackFxn ========
 *  Pin interrupt Callback function board INTB configured in the pinTable.
 */
void intBCallbackFxn(PIN_Handle handle, PIN_Id pinId) {

    PPGInterruptRequest = 1;
    return;
}
```

I²C Communication Management

Pin Configuration

```
// Import I2C Driver definitions
#include <ti/drivers/I2C.h>
#include <ti/drivers/i2c/I2CCC26XX.h>
#include <ti/drivers/Power.h>
#include <ti/drivers/power/PowerCC26XX.h>

#define CC2640R2_LAUNCHXL_I2CCOUNT 1
#define CC2640R2_LAUNCHXL_I2C0_SCL0 IOID_5
#define CC2640R2_LAUNCHXL_I2C0_SDA0 IOID_6
#define CC2640R2_LAUNCHXL_I2C0 0

I2CCC26XX_Object i2cCC26xxObjects[CC2640R2_LAUNCHXL_I2CCOUNT];

const I2CCC26XX_HWAttrsV1 i2cCC26xxHWAttrs[CC2640R2_LAUNCHXL_I2CCOUNT] = {
    {
        .baseAddr    = I2C0_BASE,
        .powerMngrId = PowerCC26XX_PERIPH_I2C0,
        .intNum      = INT_I2C_IRQ,
        .intPriority = ~0,
        .swiPriority = 0,
        .sdaPin      = CC2640R2_LAUNCHXL_I2C0_SDA0,
        .sclPin      = CC2640R2_LAUNCHXL_I2C0_SCL0,
    }
};

const I2C_Config I2C_config[CC2640R2_LAUNCHXL_I2CCOUNT] = {
    {
        .fxnTablePtr = &I2CCC26XX_fxnTable,
        .object      = &i2cCC26xxObjects[CC2640R2_LAUNCHXL_I2C0],
        .hwAttrs     = &i2cCC26xxHWAttrs[CC2640R2_LAUNCHXL_I2C0]
    },
};

const uint_least8_t I2C_count = CC2640R2_LAUNCHXL_I2CCOUNT;
// Open I2C bus for usage
I2C_Handle i2cHandle;
I2C_Params params;
```

Initialization/Register Configuration

```
// Define name for an index of an I2C bus
#define PPGSENSOR 0
// PPG Sensor MAXM86161 Address
#define MAXM86161_ADDRESS 0x62 //0x62
// Read Operation
#define I2C_OP_READ 1
// Write Operation
#define I2C_OP_WRITE 0
// Interrupt Enable 1 Register Address
#define INT_ENABLE_ADDRESS 0x02
// Enable Only Proximity Interrupt
#define PROX_INT_EN 0x10
// Enable Only Proximity Interrupt and FIFO Data Ready in the Interrupt Enable 1 Register
#define INT_EN 0x50
// Interrupt Status 1 Register Address
#define INT_STATUS_ADDRESS 0x00
// LED1 Driver Register Address (for assign LED1 current)
#define LED1_PA_ADDRESS 0x23
// LED Range for LED1 Current => (00001010b)=0x0A => 4,86mA
#define LED1_PA 0x20
// LED Pilot Pa Register Address (for assign LED current)
#define LED1_PILOT_PA_ADDRESS 0x29
// LED Range for LED1_PILOT Current => (00001010b)=0x0A => 4,86mA
#define LED1_PILOT_PA 0x20
// LED2 Driver Register Address (for assign LED2 current)
#define LED2_PA_ADDRESS 0x24
// LED Range for LED2 Current => (00001010b)=0x0A => 4,86mA
#define LED2_PA 0x20
// LED3 Driver Register Address (for assign LED3 current)
#define LED3_PA_ADDRESS 0x25
// LED Range for LED3 Current => (00001010b)=0x0A => 4,86mA
#define LED3_PA 0x20
// LED Range Register Address (for assign LED current)
#define LEDX_RGE_ADDRESS 0x2A
// LED Range for LED1(11=>124mA), LED2(11=>124mA) and LED3(11=>124mA) => (00111111b)=0x3F
#define LED_RANGE 0x3F
// LED Sequence 1-2 Control Register Address
#define LED_SEQ_CONTROL_1_ADDRESS 0x20
//// SEQ1 => Proximity Pilot on LED1 (1000) and SEQ2 => LED2(IR) (0010)
//#define LED_SEQ_CONTROL_1 0x28
// SEQ1 => Proximity Pilot on LED2(IR) (0010) and SEQ2 => LED2(IR) (0010)
#define LED_SEQ_CONTROL_1 0x32
// LED Sequence 3-4 Control Register Address
#define LED_SEQ_CONTROL_2_ADDRESS 0x21
// SEQ3 =>LED3(RED) (0011) and SEQ4 => NONE (0000)
#define LED_SEQ_CONTROL_2 0x00
//// SEQ3 =>LED3(RED) (0011) and SEQ4 => Ambient (1001)
//#define LED_SEQ_CONTROL_2 0x93
// LED Sequence 5-6 Control Register Address
#define LED_SEQ_CONTROL_3_ADDRESS 0x22
// SEQ5 => NONE (0000) and SEQ6 => NONE(0000)
#define LED_SEQ_CONTROL_3 0x00
//// SEQ5 => LED1(GREEN) (0001) and SEQ6 => NONE(0000)
//#define LED_SEQ_CONTROL_3 0x01
// PhotoDiode Register Address
#define PHOTODIODE_BIAS_ADDRESS 0x15
```

```
57 // Photodiode Capacitance => (0x01)=(00000001b) => 65pF
58 #define PHOTODIODE_BIAS 0x01
59 // PPG Configuration 1 Register Address
60 #define PPG_CONFIGURATION_1_ADDRESS 0x11
61 // (0x08)=(00001000b) => PPG1_ADC_RGE => (10) => 16uA and PPG_TINT => (00) => 14,8us
62 #define PPG_CONF_1 0x0B
63 // PPG Configuration 2 Register Address
64 #define PPG_CONFIGURATION_2_ADDRESS 0x12
65 // (0x80)=(10000000b) => PPG_SR => (10000) => 512sps and SMP_AVE => (000) => (1 => no ave)
66 #define PPG_CONF_2 0x00
67 // PPG Configuration 3 Register Address
68 #define PPG_CONFIGURATION_3_ADDRESS 0x13
69 // (0xC0)=(11000000b) => LED_SETLNG => (11) => 12us
70 #define PPG_CONF_3 0xC0
71 // Proximity Set Interrupt Threshold Register Address
72 #define PROX_INT_THRESH_ADDRESS 0x14
73 // Proximity Set Interrupt Threshold to (0x40)
74 #define PROX_INT_THRESH 0x40
75 // System Control Register Address
76 #define SYSTEM_CONTROL_ADDRESS 0x0D
77 // Soft Reset Command for MAXM86161
78 #define SYSTEM_CONTROL_RESET 0x01
79 // Mask to Interrupt Register for Proximity Sensor
80 #define MASK_PROX 0x10
81 // Mask to Interrupt Register for One Data on The FIFO
82 #define MASK_FIFO 0x40
83 // Mask to Interrupt Register for One Data on The FIFO and Proximity Sensor
84 #define MASK_FIFO_PROX 0x50
85 // FIFO Configuration 2 Register Address
86 #define FIFO_CONFIGURATION_2_ADDRESS 0x0A
87 // FIFO Flush Command
88 #define FIFO_FLUSH 0x10
89 // FIFO Data Counter Register Address
90 #define FIFO_DATA_COUNTER_ADDRESS 0x07
91 // FIFO Data Register Address
92 #define FIFO_DATA_ADDRESS 0x08
93 // Mask to Data TAG
94 #define MASK_TAG 0xF8
95
96 ...
97
98 static void I2C_Initialization(void);
99
100 ...
101
102 static void I2C_Initialization(void){
103
104     uint8_t writeBuffer1[1];
105     uint8_t writeBuffer2[2];
106     uint8_t writeBuffer3[3];
107     uint8_t writeBuffer4[4];
108     uint8_t writeBuffer7[7];
109     uint8_t readBuffer1[1];
110
111     // One-time init of I2C driver
112     I2C_init();
113     // initialize optional I2C bus parameters
114     I2C_Params_init(&params);
```

```
115     params.bitRate = I2C_400kHz;
116
117     // Open I2C bus for usage
118     i2cHandle = I2C_open(PPGSENSOR, &params);
119
120     // Initialize slave address of transaction
121     I2C_Transaction transaction = {0};
122
123     //---------------------------------------------------------------------------
124     // Soft Reset
125     //---------------------------------------------------------------------------
126
127     writeBuffer2[0] = SYSTEM_CONTROL_ADDRESS;      // System Control Register Address
128     writeBuffer2[1] = SYSTEM_CONTROL_RESET;        // Enable Soft Reset
129     transaction.slaveAddress = MAXM86161_ADDRESS;   // Slave ID for Write
130     transaction.writeBuf = writeBuffer2;
131     transaction.writeCount = 2;
132     transaction.readBuf = NULL;
133     transaction.readCount = 0;
134     I2C_transfer(i2cHandle, &transaction);
135
136     //---------------------------------------------------------------------------
137     // Set Register For Proximity Function
138     //---------------------------------------------------------------------------
139
140     writeBuffer2[0] = INT_ENABLE_ADDRESS;           // Interrupt Enable 1 Address
141     writeBuffer2[1] = PROX_INT_EN;                  // Enable ONLY Proximity Interrupt and
        FIFO Data Ready
142     transaction.slaveAddress = MAXM86161_ADDRESS;   // Slave ID for Write
143     transaction.writeBuf = writeBuffer2;
144     transaction.writeCount = 2;
145     transaction.readBuf = NULL;
146     transaction.readCount = 0;
147     I2C_transfer(i2cHandle, &transaction);
148
149     writeBuffer2[0] = PROX_INT_THRESH_ADDRESS;      // Proximity Interrupt Threshold Address
150     writeBuffer2[1] = PROX_INT_THRESH;              // Set the Proximity Threshold to 128
151     transaction.slaveAddress = MAXM86161_ADDRESS;   // Slave ID for Write
152     transaction.writeBuf = writeBuffer2;
153     transaction.writeCount = 2;
154     transaction.readBuf = NULL;
155     transaction.readCount = 0;
156     I2C_transfer(i2cHandle, &transaction);
157
158     writeBuffer7[0] = LED_SEQ_CONTROL_1_ADDRESS;    // LED Sequence Register 1 Address (
        starting point)
159     writeBuffer7[1] = LED_SEQ_CONTROL_1;            // SEQ1 => Proximity Pilot on LED1
        (1000) and SEQ2 => LED2(IR) (1000)
160     writeBuffer7[2] = LED_SEQ_CONTROL_2;            // SEQ3 => LED3(RED) (0011) and SEQ4 =>
        Ambient (1001)
161     writeBuffer7[3] = LED_SEQ_CONTROL_3;            // SEQ5 => LED1(GREEN) (0001) and SEQ6
        => NONE(0000)
162     writeBuffer7[4] = LED1_PA;                      // LED Range for LED1 Current =>
        (00001010b)=0x0A => 4,86mA
163     writeBuffer7[5] = LED2_PA;                      // LED Range for LED2 Current =>
        (00001010b)=0x0A => 4,86mA
164     writeBuffer7[6] = LED3_PA;                      // LED Range for LED3 Current =>
        (00001010b)=0x0A => 4,86mA
```

```
165     transaction.slaveAddress = MAXM86161_ADDRESS;   // Slave ID for Multi-Byte Write
166     transaction.writeBuf = writeBuffer7;
167     transaction.writeCount = 7;
168     transaction.readBuf = NULL;
169     transaction.readCount = 0;
170     I2C_transfer(i2cHandle, &transaction);
171
172     writeBuffer3[0] = LED1_PILOT_PA_ADDRESS;        // LED1 Driver Register Address (
        starting point)
173     writeBuffer3[1] = LED1_PILOT_PA;                // LED Range for LED1_PILOT Current =>
        (00001010b)=0x0A => 4,86mA
174     writeBuffer3[2] = LED_RANGE;                    // LED Range for LED1(11=>124mA), LED2
        (11=>124mA) and LED3(11=>124mA)
175     transaction.slaveAddress = MAXM86161_ADDRESS;   // Slave ID for Multi-Byte Write
176     transaction.writeBuf = writeBuffer3;
177     transaction.writeCount = 3;
178     transaction.readBuf = NULL;
179     transaction.readCount = 0;
180     I2C_transfer(i2cHandle, &transaction);
181
182     //------------------------------------------------------------------------------
183     // Set Register For PPG Configuration
184     //------------------------------------------------------------------------------
185
186     writeBuffer4[0] = PPG_CONFIGURATION_1_ADDRESS;  // PPG Configuration 1 Register Address
        (starting point)
187     writeBuffer4[1] = PPG_CONF_1;                   // PPG1_ADC_RGE => (10) => 16uA and
        PPG_TINT => (00) => 14,8us
188     writeBuffer4[2] = PPG_CONF_2;                   // PPG_SR => (10000) => 512sps and
        SMP_AVE => (000) => (1 => no ave)
189     writeBuffer4[3] = PPG_CONF_3;                   // LED_SETLNG => (11) => 12us
190     transaction.slaveAddress = MAXM86161_ADDRESS;   // Slave ID for Multi-Byte Write
191     transaction.writeBuf = writeBuffer4;
192     transaction.writeCount = 4;
193     transaction.readBuf = NULL;
194     transaction.readCount = 0;
195     I2C_transfer(i2cHandle, &transaction);
196
197     writeBuffer2[0] = PHOTODIODE_BIAS_ADDRESS;      // PhotoDiode Register Address
198     writeBuffer2[1] = PHOTODIODE_BIAS;              // Photodiode Capacitance => (0x01)
        =(00000001b) => 65pF
199     transaction.slaveAddress = MAXM86161_ADDRESS;   // Slave ID for Write
200     transaction.writeBuf = writeBuffer2;
201     transaction.writeCount = 2;
202     transaction.readBuf = NULL;
203     transaction.readCount = 0;
204     I2C_transfer(i2cHandle, &transaction);
205
206     writeBuffer1[0] = INT_STATUS_ADDRESS;           // Interrupt Status 1 Register Address
207     transaction.slaveAddress = MAXM86161_ADDRESS;   // Slave ID for Write
208     transaction.writeBuf = writeBuffer1;
209     transaction.writeCount = 1;
210     transaction.readBuf = readBuffer1;
211     transaction.readCount = 1;
212     I2C_transfer(i2cHandle, &transaction);
213
214     writeBuffer2[0] = FIFO_CONFIGURATION_2_ADDRESS; // Interrupt Status 1 Register Address
215     writeBuffer2[1] = FIFO_FLUSH;                   // FIFO Flush Command
```

```
    transaction.slaveAddress = MAXM86161_ADDRESS;   // Slave ID for Write
    transaction.writeBuf = writeBuffer2;
    transaction.writeCount = 2;
    transaction.readBuf = NULL;
    transaction.readCount = 0;
    I2C_transfer(i2cHandle, &transaction);

    // Close I2C
    I2C_close(i2cHandle);
}
```

Measurement Management in main Task

```
...

I2C_Initialization();

...

// Application main loop
for (;;){
  if( PPGInterruptRequest == 1 )
  {
      PPGInterruptRequest = 0;

      uint8_t writeBuffer1[1];
      uint8_t readBuffer1[1];
      uint8_t writeBuffer2[2];
      uint8_t readData[3];

      memset(readData, 0, 4);

      // One-time init of I2C driver
      I2C_init();
      // initialize optional I2C bus parameters
      I2C_Params_init(&params);
      params.bitRate = I2C_400kHz;
      // Open I2C bus for usage
      i2cHandle = I2C_open(PPGSENSOR, &params);
      // Initialize slave address of transaction
      I2C_Transaction transaction = {0};

      writeBuffer1[0] = INT_STATUS_ADDRESS;         // Interrupt Status 1 Register Address
      transaction.slaveAddress = MAXM86161_ADDRESS; // Slave ID for Write
      transaction.writeBuf = writeBuffer1;
      transaction.writeCount = 1;
      transaction.readBuf = readBuffer1;
      transaction.readCount = 1;
      I2C_transfer(i2cHandle, &transaction);

      if ((readBuffer1[0] & MASK_PROX) == MASK_PROX)
      {
          if(PPGStart == 0)
          {
              writeBuffer2[0] = FIFO_CONFIGURATION_2_ADDRESS; // Interrupt Status 1 Register
     Address
              writeBuffer2[1] = FIFO_FLUSH;                   // FIFO Flush Command
              transaction.slaveAddress = MAXM86161_ADDRESS;   // Slave ID for Write
```

```
45              transaction.writeBuf = writeBuffer2;
46              transaction.writeCount = 2;
47              transaction.readBuf = NULL;
48              transaction.readCount = 0;
49              I2C_transfer(i2cHandle, &transaction);
50
51              writeBuffer2[0] = INT_ENABLE_ADDRESS;           // Interrupt Enable 1 Address
52              writeBuffer2[1] = INT_EN;                       // Enable Proximity Interrupt
    and FIFO Data Ready
53              transaction.slaveAddress = MAXM86161_ADDRESS;   // Slave ID for Write
54              transaction.writeBuf = writeBuffer2;
55              transaction.writeCount = 2;
56              transaction.readBuf = NULL;
57              transaction.readCount = 0;
58              I2C_transfer(i2cHandle, &transaction);
59          }
60          if(PPGStart == 1)
61          {
62              writeBuffer2[0] = FIFO_CONFIGURATION_2_ADDRESS; // Interrupt Status 1 Register
     Address
63              writeBuffer2[1] = FIFO_FLUSH;                   // FIFO Flush Command
64              transaction.slaveAddress = MAXM86161_ADDRESS;   // Slave ID for Write
65              transaction.writeBuf = writeBuffer2;
66              transaction.writeCount = 2;
67              transaction.readBuf = NULL;
68              transaction.readCount = 0;
69              I2C_transfer(i2cHandle, &transaction);
70
71              writeBuffer2[0] = INT_ENABLE_ADDRESS;  // Interrupt Enable 1 Address
72              writeBuffer2[1] = PROX_INT_EN;         // Enable Proximity Interrupt
73              transaction.slaveAddress = MAXM86161_ADDRESS;   // Slave ID for Write
74              transaction.writeBuf = writeBuffer2;
75              transaction.writeCount = 2;
76              transaction.readBuf = NULL;
77              transaction.readCount = 0;
78              I2C_transfer(i2cHandle, &transaction);
79          }
80
81          PPGStart = !PPGStart;
82          memset(PPGData1, 0, NUMB);
83          memset(PPGData2, 0, NUMB);
84          PPGDataNumb1 = 0;
85          PPGDataNumb2 = 0;
86          PPGDataStop = 0;
87      }
88      if (((((readBuffer1[0] & MASK_FIFO) == MASK_FIFO) && (PPGStart == 1))  && (PPGDataStop
    == 0)))
89      {
90          writeBuffer1[0] = 0x08;
91          transaction.slaveAddress = MAXM86161_ADDRESS;
92          transaction.writeBuf = writeBuffer1;
93          transaction.writeCount = 1;
94          transaction.readBuf = readBuffer1;
95          transaction.readCount = 1;
96          I2C_transfer(i2cHandle, &transaction);
97
98          for(int w = 0; w < readBuffer1[0]; w++)
99          {
```

```
100             if(PPGDataStop == 0)
101             {
102                 writeBuffer1[0] = FIFO_DATA_ADDRESS;          // FIFO Data Address
103                 transaction.slaveAddress = MAXM86161_ADDRESS; // Slave ID for Write
104                 transaction.writeBuf = writeBuffer1;
105                 transaction.writeCount = 1;
106                 transaction.readBuf = readData;
107                 transaction.readCount = 3;
108                 I2C_transfer(i2cHandle, &transaction);
109
110                 if((readData[0] & MASK_TAG) != 0b11110000)
111                 {
112                     if((readData[0] & MASK_TAG) != 0b00001000)
113                     {
114                         PPGData1[PPGDataNumb1] = (((readData[0]<<16 | readData[1]<<8) |
    readData[2]));
115                         PPGDataNumb1 = PPGDataNumb1 + 1;
116                     }
117                     if((readData[0] & MASK_TAG) != 0b00010000)
118                     {
119                         PPGData2[PPGDataNumb2] = (((readData[0]<<16 | readData[1]<<8) |
    readData[2]));
120                         PPGDataNumb2 = PPGDataNumb2 + 1;
121                     }
122                     if(PPGDataNumb1 == NUMB)
123                     {
124                         PPGDataStop = 1;
125                         writeBuffer2[0] = INT_ENABLE_ADDRESS; // Interrupt Enable 1
    Address
126                         writeBuffer2[1] = PROX_INT_EN;        // Enable ONLY Proximity
    Interrupt
127                         transaction.slaveAddress = MAXM86161_ADDRESS;  // Slave ID for
    Write
128                         transaction.writeBuf = writeBuffer2;
129                         transaction.writeCount = 2;
130                         transaction.readBuf = NULL;
131                         transaction.readCount = 0;
132                         I2C_transfer(i2cHandle, &transaction);
133                     }
134                 }
135             }
136             else
137             {
138                 break;
139             }
140         }
141         Prova += 1;
142     }
143     // Close I2C
144     I2C_close(i2cHandle);
145 }
146
147 ...
148
149 }
```

Part III

Testing and Conclusion

Chapter 6

Testing

This chapter is dedicated to testing. Some tests were done at the beginning of the project to understand how the OLD ECG worked, while other tests were carried out at the end to understand if the device designed and described in the previous chapters worked. Below are reported some illustrative images of the tests carried out. They have been divided into two categories:

- tests carried out with laboratory instruments such as the oscilloscope;
- tests carried out with the use of the tablet, then through the use of a dedicated app.

6.1 Old ECG Tests

The initial test that has been performed concern the OLD ECG. As it can be seen in the images below, both through the oscilloscope and through the app you can notice the disturbances of the electrical network.

Figure 6.1. OLD ECG Signal from Oscilloscope.

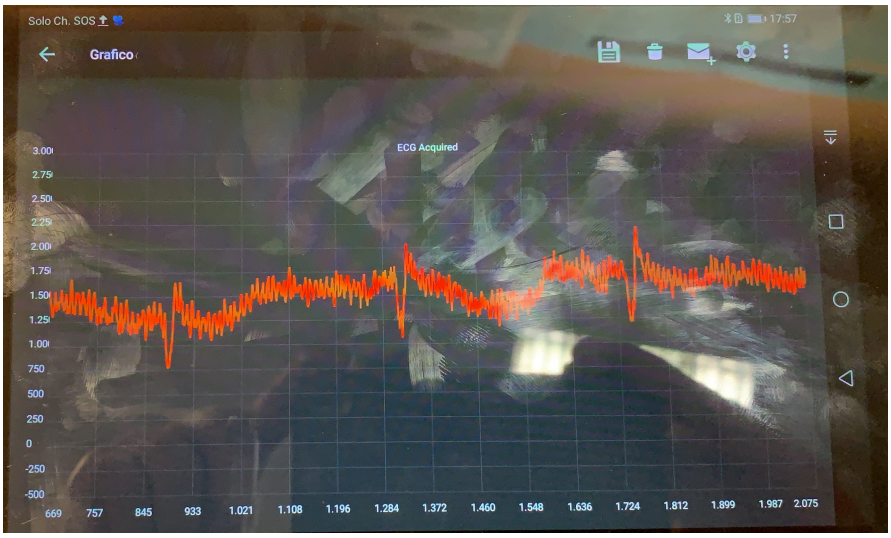


Figure 6.2. OLD ECG Signal from App.

6.2 New ECG and PPG Tests

In this section has been inserted three important test done for the verification of the proper operation of the new device. It has been perform three test:

- I^2C verification;
- ECG verification from App;
- PPG verification from Matlab.

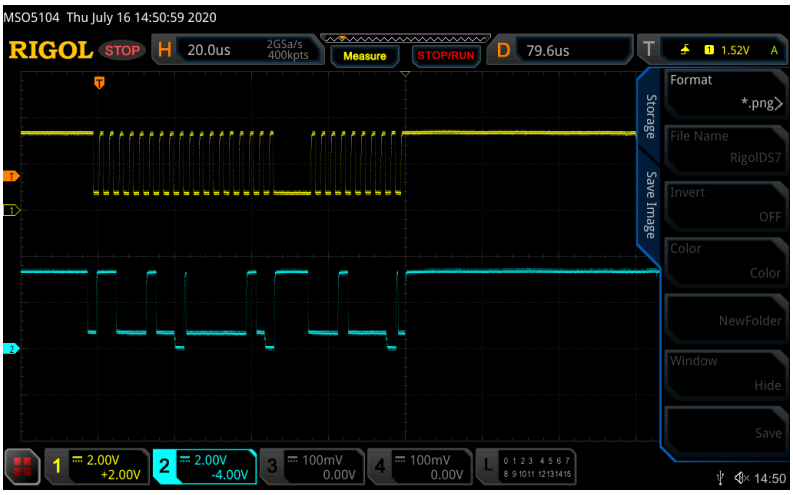


Figure 6.3. I^2C Communication from Oscilloscope.

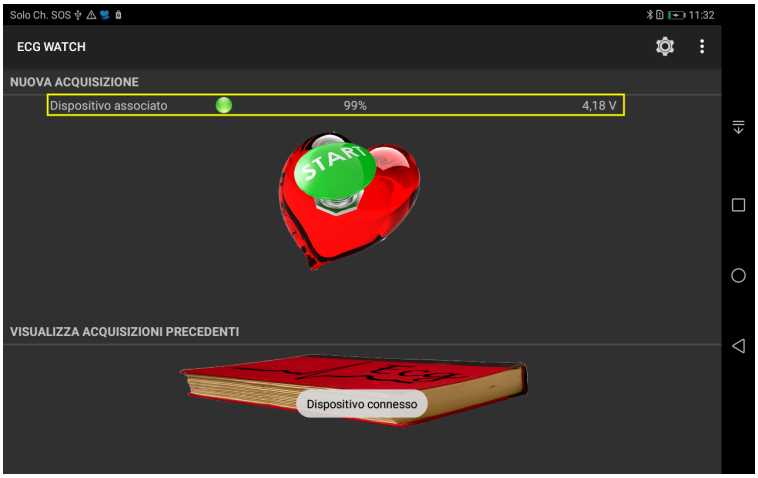


Figure 6.4. I^2C Battery Communication from App.

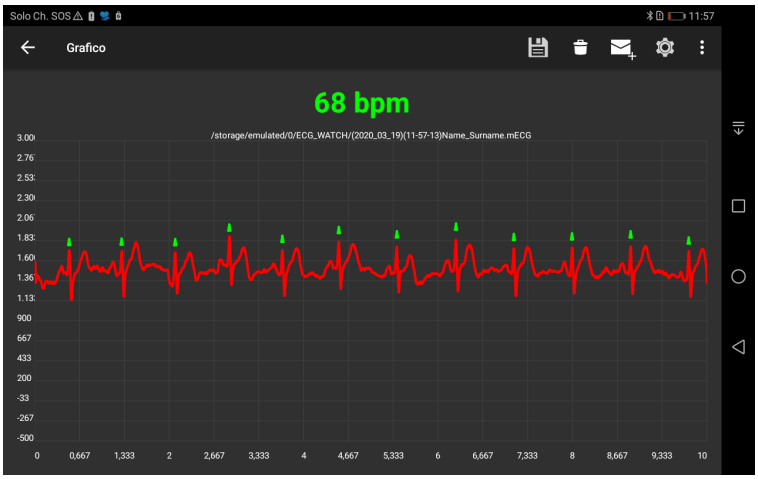


Figure 6.5. ECG Signal from App.

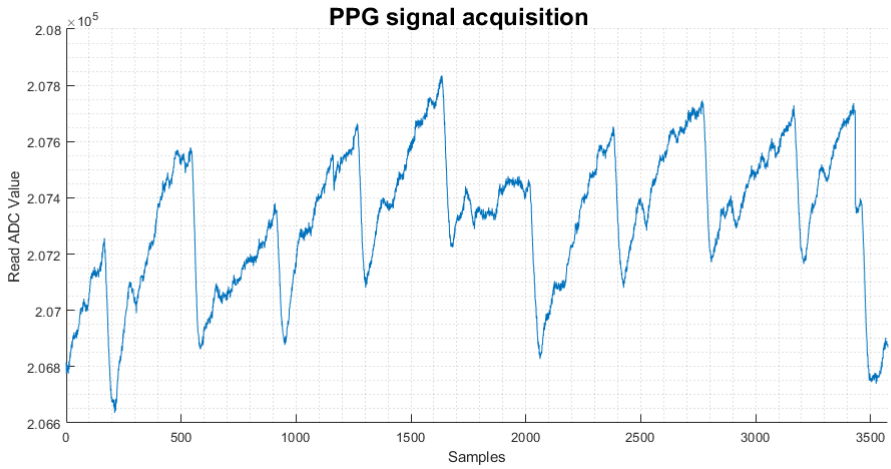


Figure 6.6. PPG Signals fro MATLAB.

Chapter 7

Future Perspectives and Conclusions

7.1 Future Perspectives

- **Future Hardware developments:** some other sensors can be implemented in order to make it more complex such as temperature sensors, motion sensors and other health sensors. An other improvement can be the introduction a chip antenna for BLE communication in order to reduce more the device dimension. Another improvement can be the addition of Flash memory for saving data on board before being released to the app.
- **Future Firmware developments:** firmware can be cleaned in order to make the code more readable, also a new characteristic could be added to send the PPG data from the device to the app.
- **Future App developments:** for App could be useful to display the PPG value and, through other algorithms, the calculation of blood pressure with a non-invasive method.

7.2 Conclusion

The aim of this thesis was to create an object that was able to measure some vital parameters such as ECG and SpO $_2$. The results obtained have been quite satisfactory although something will, certainly, have to be reviewed as presented in the Future Perspectives section.

In this pandemic period, caused by the arrival of a virus called *Covid-19*, such a device can be a tool for an initial analysis of a patient's condition or it can be a tool for a daily check of vital signs.

To sum up, the requirements have been quite satisfied, this object can be a good starting point for the creation of other useful objects for monitoring the life quality.

Bibliography

[1] Zhaksylyk Kudaibergenov and Talgat Bekkaliyev, " Twin-T Notch Active Filter", Abstract, Nazarbayev University, 2016

[2] HankZ., "Twin T Notch Filter", AnalogDevices, MiniTutorial, 2012. [Online]. Available: https://www.analog.com/media/en/training-seminars/tutorials/MT-225.pdf, [Accessed Jul. 11, 2020].

[3] http://sim.okawa-denshi.jp/en/TwinTCRkeisan.htm, [Accessed Jul. 11, 2020].

[4] http://sim.okawa-denshi.jp/images/CRTwinTD.gif, [Accessed Jul. 11, 2020]

[5] Jim Karki, "Active Low-Pass Filter Design", Texas Instrument, Application Report, SLOA049B - September 2002.

[6] Texas Instruments, "SimpleLink^{TM} Bluetooth®low energy Wireless MCU for Automotive", CC2640 Datasheet, SWRS176B, February 2015 [Revised July 2016].

[7] Maxim Integrated Products, "Buck/Boost Regulating Charge Pump in µMAX", MAX1759 Datasheet, Rev 1, 2000.

[8] Texas Instruments, "REF20xx Low-Drift, Low-Power, Dual-Output, VREF and VREF / 2 Voltage References", REF2025, REF2030, REF2033, REF2041 Datasheet, SBOS600D - May 2014 – Revised July 2018.

[9] Maxim Integrated Products, "SOT23 Dual-Input USB/AC Adapter 1-Cell Li+ Battery Chargers", MAX1551/MAX1555 Datasheet, Rev 0, 2003.

[10] Maxim Integrated Products, "3 µA 1-Cell/2-Cell Fuel Gauge with ModelGauge", MAX17048/MAX17049 Datasheet, Rev 7, 2016.

[11] STMicroelectronics, "Ultra low capacitance ESD protection", DVIULC6-2x6 Datasheet, Rev 2, October 2015.

[12] Texas Instrument, "INA333 Micro-Power (50µA), Zerø-Drift, Rail-to-Rail Out Instrumentation Amplifier" INA333 datasheet (Rev. C), December 2015.

[13] Texas Instrument, "OPAx33050-µVVOS, 0.25-µV/°C, 35-µA CMOS Operational Amplifiers Zero-Drift Series", OPA330, OPA2330, OPA4330 Datasheet, SBOS432G - August 2008 - Revised August 2016.

[14] Maxim Integrated Products, "Single-Supply Integrated Optical Module for HR and SpO2 Measurement", MAXM86161 Datasheet, Rev 0, 2019.

[15] JST Connector, "1.0mm pitch/Disconnectableb Crimp style connectors", SM02B-SRSS-TB(LF)(SN) Datasheet.

[16] Lite-On Inc., "SMD LED LTST-C193KRKT-5A", LED Red Datasheet, 03 May 2020.

[17] Panasonic, "3.5 mm×2.9 mm Side-operational SMD Light Touch Switches", Light Touch Switches/EVQP7/P3/9P7 Datasheet, March 2019.

[18] Samtec, "THROUGH-HOLE MICRO HEADER", JTAG Datasheet, F-219 (Rev 11 May 20).

[19] Maxim Integrated Products, "Recommended Configurations and Operating Profiles for MAX30101/MAX30102 EV Kits", UG6409, Rev 0, March 2018.

[20] Rusch, T. L., Sankar, R., Scharf, J. E. (1996). "Signal processing methods for pulse oximetry. Computers in biology and medicine", 26(2), 143-159.

[21] J. G. Webster, "Design of Pulse Oximeters", Series in Medical Physics and Biomedical Engineering, Taylor Francis, New York, USA, 1997