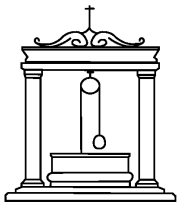
SAPIENZA
UNIVERSITÀ DI ROMA

Sapienza University of Rome

Faculty of Information Engineering, Informatics, and Statistics
Department of Information Engineering, Electronics,
and Telecommunications



Portable Devices for Bio-Sensing And Impedance Analysis

Bruno Donato

Supervisors:
Dr. Sara Ghoreishizadeh
Prof. Fernanda Irrera

Coordinator:
Dr. Pantelis Georgiou

Academic Year 2017-18

Abstract

This work concerns the realisation of a portable device for bio-sensing and impedance analysis, and has been carried out in its entirety at the *"Centre for Bio-Inspired Technology", Imperial College London (UK).*

In the field of *bio-sensing*, electronic devices allow the detection of chemical indicators in biological or body fluids. The development of such devices is driven by the will of allowing rapid and cost-effective *bio-markers* measurements in *medical diagnostics* applications, both in industry and in academia. To achieve this, different *sensing techniques* are employed, depending on the target analyte. These, by electrically stimulating a sample solution, generate a reaction which can be correlated with the presence of the compound of interest. One promising sensing method is *Electrochemical Impedance Spectroscopy (EIS).* Affirmed since the mid-1980s as electroanalytical method, interest in EIS grew afresh recently, following the discovery of new medical applications such as bacteria sensing, cancer markers detection, DNA sensing, and hormones monitoring. Indeed, the technique revealed many advantages with respect to conventional diagnostic methods (e.g. *polymerase chain reaction (PCR)*, *enzyme linkedimmunosorbent assay (ELISA)*), like rapid response and low detection limits. Amongst other sensing techniques are *Chronoamperometry (CA)* and *Cyclic Voltammetry (CV)*, well know *electroanalytical methods* for studying analytes' presence in a solution by triggering *Reduction-Oxidation (RedOx)* reactions.

Realising portable devices for performing this types of measurements enables peripatetic personal analysis of samples, allowing users to monitor their body fluids and gather vital information for diagnostic or therapeutic uses. An example is that one of glucometers, devices used by diabetic patients to check glucose levels in their blood.

The aim of this work is to develop a portable device for performing EIS, CA, and CV. For achieving this, three steps have been executed: (i) realisation of a prototype platform, (ii) conception and design of a portable *Printed Circuit Board (PCB)*, (iii) testing and comparison of the two devices. In addition, an important feature of the final device is the ability of functioning without a battery. This objective has been accomplished using *hybrid energy harvesting*, powered by *Near Field Communication (NFC)* and a piezoelectric harvester.

Acknowledgements

The completion of this work could not have been possible without the contribution and the assistance of so many people that names may not all be enumerated. Yet, I would like to express my deep gratitude particularly to the following:

- My supervisor at the "Center for Bio-Inspired Technology (CBIT)", **Dr. Sara Ghoreishizadeh**, for being of essential assistance, daily, for the whole duration of my project, and for proposing me to work with her in this field, giving me the opportunity of gaining knowledge in such a multitude of subjects.

- My supervisor at "Sapienza University of Rome", **Prof. Fernanda Irrera**, for remotely coordinating and supervising me, and for first instilling me the desire to learn more about the new applications of electronics and CMOS technologies in other fields.

- My coordinator at "Imperial College London", **Dr. Pantelis Georgiou**, for accepting me in his amazing and greatly talented research group, and providing me with the opportunity of learning from this experience more than I could ever imagine.

- **Prof. Sandro Carrara**, for first introducing me to the fascinating world of electrochemical sensing, and for making me dream about future biological and medical applications of electronics.

- **All students and researchers at CBIT** for the helpful discussions, for the answers they gave to my numerous questions, and for the warmth with which they welcomed me in their research group. A special remark to my friend and colleague **Dorian Haci**, for the continuous support he provided me with throughout the different phases of my project, particularly in the design and realisation of the PCB device.

- Last, my immense gratitude goes to **my family and friends**, that gave their support morally, financially and physically in my journey through university and life.

Contents

List of Tables

List of Figures

Introduction

Motivation

Since its birth electronics has developed alongside all other fields of study, leading to the discovery of new means of communication, new tools for enhancing our capacities and new ways for entertaining ourselves. A plethora of devices gave humans enhanced and ameliorated lives, changing many aspects of our existence. More recently, the creation of personal and portable devices led to the beginning of a deeper relationship between electronics and its users. Indeed, some of the electronic devices we use often become an essential tool for our everyday activities. Nowadays, the evolution of electronics led to the formation of an even stronger bond with humans, making their life sometimes dependable on electronics itself.

In the field known as bio-electronics, engineers are able to create devices capable of interacting directly with biology and biochemistry, for both diagnostic and therapeutic applications. Uses of bio-electronics today help to treat or prevent human diseases or conditions, e.g. the use of cochlear implants to help restore the sense of sound in patients with severe hearing loss, or the use of glucose sensors and insulin pumps for treating type 1 diabetes conditions. Many bio-electronic applications contain one or more ways to get information from the biological world, either in the form of bio-signals (e.g. heart-beat, brain waves, etc.) or of bio-chemical information. This latter category is examined by a branch called bio-sensing, that aims to investigate the presence of atoms, molecules or compounds of interest (i.e. bio-markers) in biological or body fluids (e.g. blood, saliva, sweat, etc.). Therefore, this introduces the challenge of interfacing electronics with aqueous solutions. Different methods are used for gathering bio-markers information. These vary in terms of interface (i.e. type of electrdeode an sensor) and in the

stimulation and sensing technique.

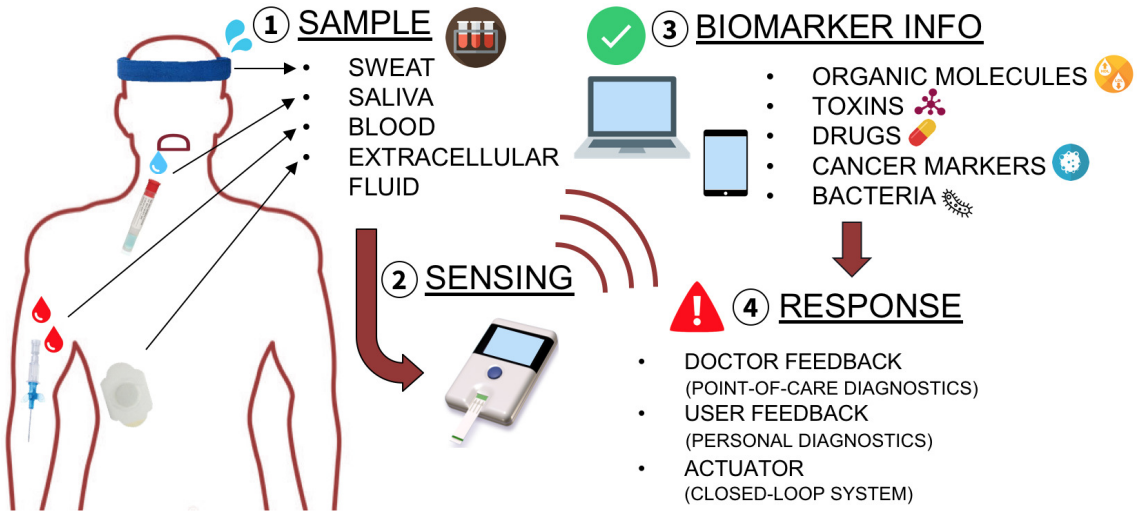


Figure 1: Illustration showing the principle of bio-sensing

In this work, three types bio-sensing technique have been examined: Chronoamperometry (CA), Cyclic Voltammetry (CV), and Electrochemical Impedance Spectroscopy (EIS).

With regard to EIS, the wide range of novel uses the technique has in the area of medical diagnostics have been driving the motivation of this project. Examples range from bacteria [1] to DNA detection [2], hormones monitoring [3], cancer markers detection [4], human proteins sensing [5], and also for the analysis of sensors' performances [6]. Further, in other fields, impedance's spectrum analysis has been used lately for many different aims. Remarkable examples are: batteries testing [7], food analysis [8], corrosion monitoring [9], and fuel cells characterisation [10]. Nevertheless, EIS measurements are rather often conducted in laboratory environment, using bulky and expensive instrumentation; indeed, not many have been so far the studies attempting to make this type of measurement portable and affordable. In medical diagnostics, EIS has the potential of allowing sensing of various analytes with many advantages with respect traditional laboratory methods (e.g. polymerase chain reaction (PCR), enzyme linked immunosorbent assay (ELISA)): faster measurement, low detection limits, cost-effectiveness, and possibility of performing real-time monitoring of samples [11, 12]. Furthermore, EIS has been recently found useful also in the detection of hormones [3], which may lead to interesting applications in personal portable and wearable diagnostics (e.g. for continuous monitoring of

hormones levels in professional athletes).

On the other hand, CA and CV electroanalytical methods have been considered. These types of measurements, very well known in chemistry laboratories, have been applied so far to a countless number of sensing applications, often related to medical fields. A notorious case is the one of glucose sensing, used in diagnostic devices that are able to reveal glucose content in a blood sample thanks to a bio-electronic interface [13]. Among other interesting endogenous indicators are: lactate, glutamate and cholesterol, all measurable with the two mentioned methods [14]. Other possible uses in medical applications are for exogenous compounds sensing, such as continuous monitoring of drugs (e.g. monitoring of analgesics [15], anaesthetics [16], anti-inflammatory compounds [17]).

Objectives

The aim of the project is to examine possible ways of designing a portable device for electrochemical and impedance measurements, using off-the-shelf components. Moreover, to explore the limitations of different implementations (i.e. development boards setup vs custom PCB design) and the possibility of minimising the power consumption of such a device to realise a battery-less PCB for portable point-of-care or personal diagnostics.

In order to achieve this goal, an action plan has been developed as follows:

- Review the state-of-the-art of portable bio-sensing devices and energy harvesting;
- Build a prototype setup for electrochemical measurements using development boards;
- Develop firmware and software for running measurements with the prototype platform;
- Design a final prototype PCB for electrochemical sensing and impedance analysis;
- Develop firmware for running measurements and transferring results;
- Test and perform measurements with the two setups.

Report Outline

- To begin with, in *chapter 1* a brief introduction to key theoretical concepts about bio-sensing will be presented. These are essential for the understanding of further design

choices and discussion. Also, a review of the state-of-the-art of EIS devices will be shown.

- Then, in *chapter 2*, a summary of energy harvesting sources will be presented, and scavenging methods for portable devices will be described. Moreover, some applications of this methods in literature, that have been taken into account for this work, will be discussed.

- *Chapter 3* will then describe in detail the work done in the prototyping stage, focusing on design choices, setup features and results of the setups' testing. Also, measurement results will be shown and system's limitation will be discussed.

- Next, *Chapter 4* will focus on the conception, design, realisation and testing of the PCB.

- Finally, conclusions will be drawn with a summary of all project's accomplishments, and a possible evolution of this work will be proposed.

Chapter 1

Bio-Sensing

The aim of bio-sensing is detecting precisely, in a given solution sample, indicators of a certain biological state or condition, commonly called *bio-markers*. These are generally found in electrolytic solutions which, thanks to their propriety of being conductive, are key for interfacing with electronics. Indeed, using electrodes and proper circuit configurations (i.e. a *Bio/CMOS interface*) a sample can be stimulated in order to induce a chemical or physical reactions correlated with the presence of a target bio-marker. Many reactions often happen simultaneously,

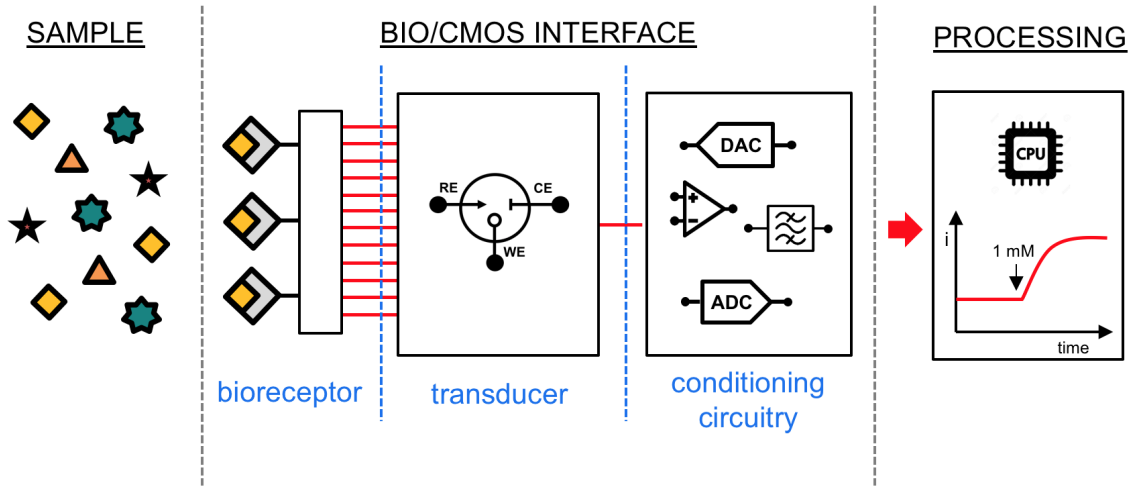


Figure 1.1: Block diagram of a typical Bio/CMOS interface

both due to the presence of multiple analytes in the medium or to chemical phenomena happening at the bio-electronic interface. Using the right type of electrochemical measurement

and the right Bio/CMOS interface is crucial to achieve specificity in the detection of a target compound. Finally, with the help of processing units, information about the presence of a specific bio-marker (e.g. the concentration of an analyte) can be extracted.

1.1 Basic Definitions

1.1.1 Bio-markers

In bio-sensing the word "bio-marker" refers to any type of measurable indicator in a biological sample. An indicator can be either a simple molecule, or a protein, or a gene. Bio-markers can be divided in two main categories: *endogenous*, if generated within the human body (e.g. hormones), and *exogenous* (e.g. drugs), if originated externally. These are linked to two types of application: therapeutic, e.g. metabolites monitoring, drugs monitoring, or diagnostic, e.g. cholesterol sensing, bacteria detection. Examples of the two types of bio-markers are shown in 1.1, in brackets the diagnostics field or the disease they are involved in.

Bio-markers	
Endogenous	*Exogenous*
Glucose (diabetes)	Escherichia coli (bacteria detection) [1]
Cholesterol (cardiovascular diseases)	Propofol (anaesthetics monitoring) [16]
Cortisol (hormone monitoring) [3]	Salmonella (bacteria detection)
PSA (prostate diagnostics)	H5N1 (avian influenza virus detection) [18]

Table 1.1: Examples of targeted bio-markers and compounds

Since the interest is often in gathering information about the presence of one single bio-marker at the time (i.e. per Bio/CMOS interface), usually we refer to bio-markers as *target molecules* or simply *targets*. We call then *specific interactions* all phenomena happening at the Bio/CMOS interface due to the presence of our target, and *nonspecific interactions* all the others. Indeed, the presence of non specific molecules creates background electrical signals: what is called *"biological noise"*. This happens due to the presence of charges that are carried along certain molecules themselves, which, as released, create a noise current at the Bio/CMOS in-

terface.

The detection of a target molecules can be achieved using different methods and techniques, which will be further discussed in the next sections. In brief, in all cases described in this work, a bio-marker is sensed applying a stimulating voltage to the Bio/CMOS interface and measuring a current response. This current carries information about the concentration of bio-marker in the sample, as well as biological noise. In some conditions, current is not only caused by the applied voltage, but also generated within the medium. This occurs when an analyte loses or gains electrons (i.e. oxidises or reduces) in response to an applied voltage, in other words when RedOx reactions take place in the solution.

1.1.2 Three-electrodes Cell

For managing the interface between solution and electronics a minimum of three electrodes are required: (i) a working electrode (WE), on which we want the reaction to take place, (ii) a reference electrode (RE), used to apply a potential across the electrochemical interface, and (iii) a counter electrode (CE), for measuring the result of the reaction, typically by means of a current flowing through the WE. This configuration is called three-electrodes cell or electrochemical cell, and is usually represented with a symbol as in 1.2. The reason why two electrodes are

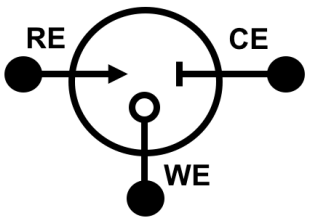


Figure 1.2: Electrochemical/three-electrode cell symbol

not enough for electrochemical measurements lies in the nature of the electrochemical medium itself. Indeed, an electrolytic solution is a conductive medium, thus presents a relative conductivity. Using only two electrodes for biasing the solution, the voltage at the working electrode interface would be different than the one applied to the reference, due to the solution's resistiv-

ity. As in 1.3: $V_z = V_o - V_s$, with V_z voltage across the interface and V_o applied to the reference electrode. Since $V_s = R_s * I_R$, with I_R being the current flowing through the cell, for nullifying

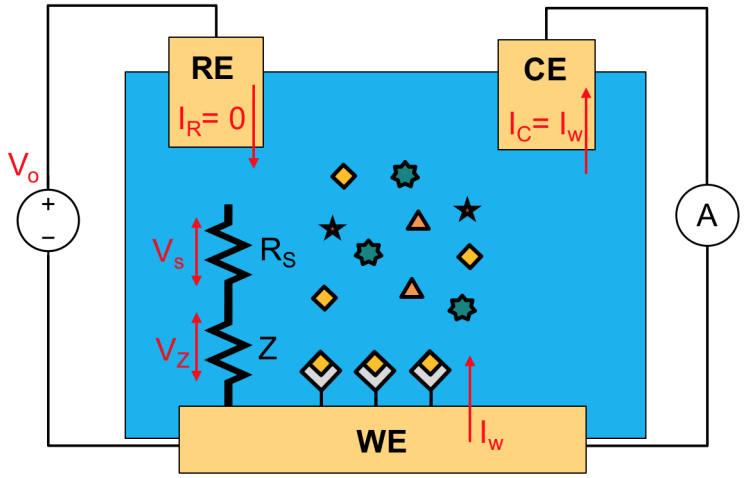


Figure 1.3: Biasing scheme of an electrochemical cell

the voltage drop the current needs to tend to zero (i.e. no current flow). This can be achieved only decoupling the circuit for applying the potential from the one sensing the output current, using a third electrode (the counter electrode).

The three electrodes can be made of different materials and in different sizes. Tuning this parameters changes the proprieties of a sensor (e.g. sensitivity, current range, equivalent capacitance, etc). For the design of the biasing electronics it is essential to know this properties of the type of interface one needs to use. For this, as will be further discussed in the next section, equivalent models are employed.

1.1.3 Electrode Functionalization

For obtaining specificity, for triggering a reaction in the solution, or for enhancing electrons transfer, chemical nano-structures at the interface are often extensively employed. The process of building this functional layers is called *"electrode functionalization"*. For instance, protein chains can be assembled on the surface of an electrode for creating a "socket" to which a target molecule can bond, or enzymes can be deposited on the electrode to select and be sensitive to a specific analyte (e.g. glucose, a drug, etc.).

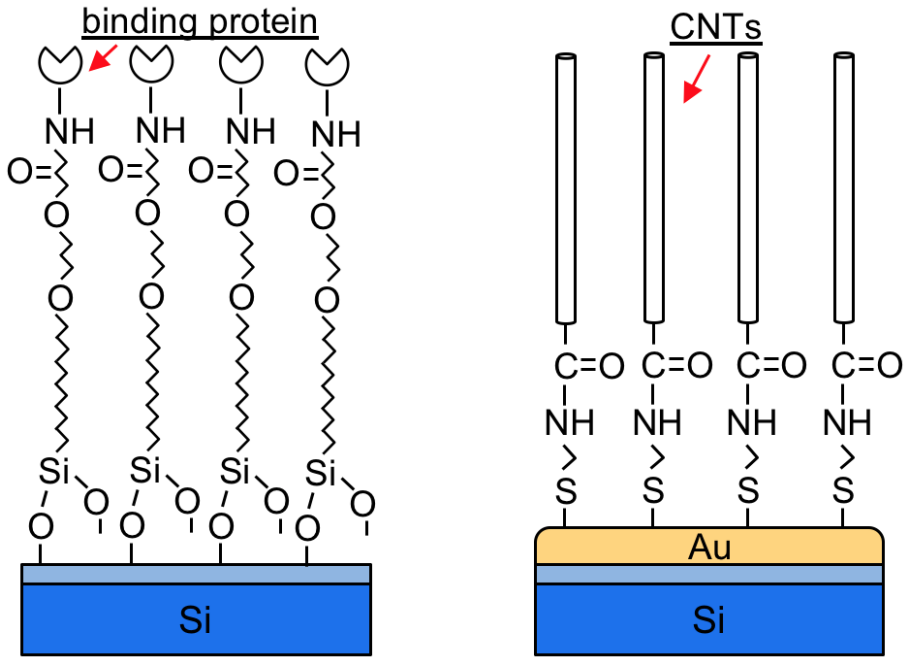


Figure 1.4: Two examples of electrode functionalization: left, assemble for protein sensing, right, enzime based functional layer

Grasping this mechanisms is important for understanding how small particles can be detected by electronics, although, all chemical procedures used for preparing electrodes for a specific target molecule are often carried on by specialists, rather than by the designer of the electronics for sensing. Therefore, for the purpose of this work, electrode functionalization will not be explained in more details [1].

1.2 Equivalent Models

In designing a Bio/CMOS interface, the electrochemical cell is the end-point of our electronics, in other words the system's load. As any kind of load, we need to know its properties before designing the electronics to interface it with. Nevertheless, an electrochemical interface is not as simple as a single discrete component. Thus, models are used in order to derive an electrical equivalent of the electrochemical junction. Having an accurate model is essential for designing and simulating a bio-sensing front-end with computer-aided design (CAD) tools.

[1] Interested readers can refer to "S.Carrara, *Bio/CMOS Interfaces and Co-Design*, 2013" [19]

1.2.1 Randles Model

The equivalent model for a polarized electrode in an electrolytic solution is composed by a resistance (R_S) in series with a parallel R-C ($R_L//C_{DL}$, as shown in 1.5). This structure is called *Randles circuit* (or Randles model). R_S is the solution resistance described before. The R_L-C_{DL} parallel instead represents the impedance of the electrode/solution interface (Z in 1.3). This representation derives from the physical phenomena happening at the interface. Current

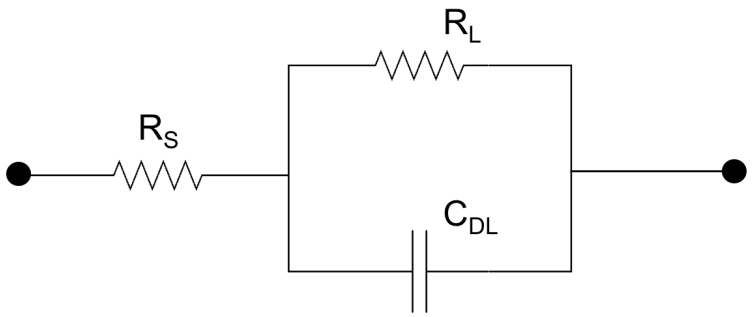


Figure 1.5: Equivalent model for polarized electrode

flowing through the electrolyte creates a charge displacement in the form of planes parallel to the electrode's surface (*Helmholtz planes*). This layering effect creates a so-called *double-layer capacitance*, which contributes to the capacitive behaviour of the impedance. Thus, the electrical impedance of the Bio/CMOS interface can be written as:

$$Z = \frac{R_L}{j\omega C_{DL} R_L + 1} \tag{1.1}$$

Whenever the interest is in monitoring a change of the impedance itself (e.g. in capacitive detection of analytes) the Randles model is often modified using a so called *constant phase element (CPE)* instead of C_{DL} (shown in 1.6).The CPE is used in electrochemistry for describing nonideal capacitance behaviour and to take into account the change of impedance with frequency.

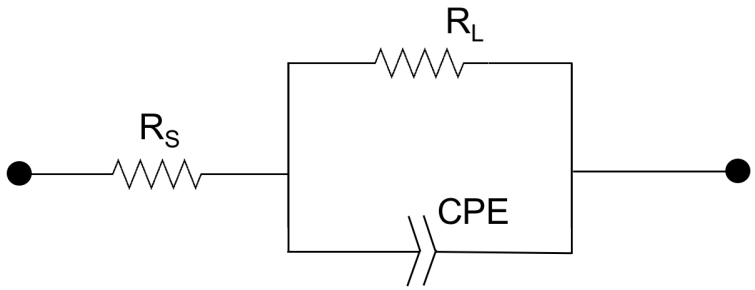


Figure 1.6: Randles model with constant phase element

It's impedance is defined as:

$$Z_{CPE} = \frac{1}{C(j\omega)^{\alpha}} = \begin{cases} R_{CPE} = \frac{1}{\omega^{\alpha} C}\sqrt{1-\alpha^2} \\ X_{CPE} = \frac{1}{\omega^{\alpha} C}\alpha \end{cases} \tag{1.2}$$

where R_{CPE} and X_{CPE} are the resistance and the reactance of the CPE respectively, ω the angular frequency, and α is a parameter between 0 and 1 (case $\alpha = 1$ corresponds to a standard capacitor). Usually, in capacitive measurements, the CPE's impedance value can be measured in frequency, and the spectral behaviour is then correlated to the presence of an analyte in the solution. This principle is what Electrochemical Impedance Spectroscopy (EIS) is based on.

1.2.2 Three-electrode Models

Regarding the three-electrode cell, two equivalent models are generally used: a *passive model*, in absence of generated currents, and an *active model*, for taking into account the reactions happening in the liquid medium. As already said, the stimulation of an electrochemical cell triggers chemical reactions that generate electrons (i.e. current) when certain conditions are met. The passive model (shown in 1.7) considers only layering effects happening when an electrode is immersed in a solution. This is valid for both the WE and the CE, whereas for the RE, in which current should not flow, the layering can be considered negligible. The resistors R_{SW}, R_{SR}, R_{WC} represent the solution resistivity at the three interfaces calculated using Ohm's second law:

$$R_S = \rho_S \frac{d}{A} \tag{1.3}$$

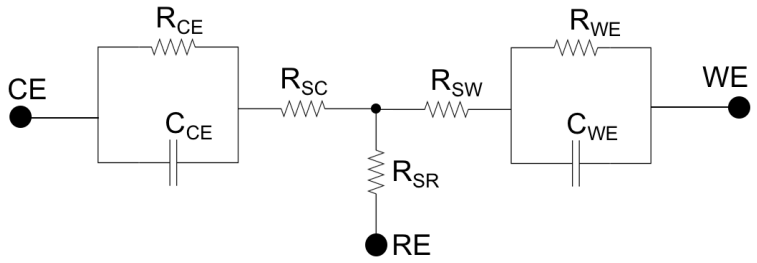


Figure 1.7: Passive model of a three-electrodes cell

with ρ_S being the solution resistivity, d the distance between electrodes, and A the area of the electrode. The passive model is used for CAD simulations in the analysis of noise and stability, but is not enough for evaluating the dynamic behaviour of the cell. For this, an active model is employed (in 1.8), that adds in the equivalent circuit current generators taking into account the action of chemical reactions in the solution. For RedOx reactions, which are responsible for the generation of faradaic currents, $i_F(t)$ is added. For all non-faradaic currents an current generator I_{NF} is introduced. The amount of current this generators represent are calculated

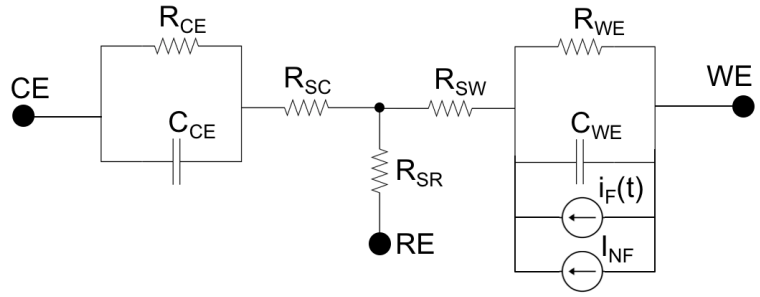


Figure 1.8: Active model of a three-electrodes cell

using different equations depending on the type of reaction. For RedOx reactions, i.e. faradaic currents, *Cotrell Equation* is used. This gives a value of $i_F(t)$ with respect to the analyte's concentration:

$$i_F(t) = \frac{nFA\sqrt{D}C(t)}{\sqrt{\pi t}} \tag{1.4}$$

where n is the number of electrons involved in the chemical reaction, F is the *Faraday constant*, A the area of the electrode, D the *diffusion coefficient* (Fick's Law), and t the time. I_{NF} value,

related to all non RedOx processes, is instead evaluated only with Randles model:

$$I_{NF} = \frac{V_{ref}}{Z} = \frac{j\omega C_{DL} R_L + 1}{R_L} V_{ref} \tag{1.5}$$

with V_{ref} applied potential, C_{DL} and R_L capacitance and resistance from Randles model.

1.3 Sensors Instrumentation and AFE

To meet the conditions for performing an electrochemical measurement electronics needs to be properly employed (or designed) for managing the three-electrode cell. As we said, a constant bias has to be applied to it, while sensing the flow of generated current. For achieving this, so-called *potentiostatic* circuits are used.

Two are the simplest, and most commonly used, CMOS configurations for bio-sensing: (i) *Grounded Counter Electrode*, and (ii) *Grounded Working Electrode*. In the first, a biasing voltage is applied between WE and RE and current flows between WE and CE. In terms of circuitry, this is usually achieved using a voltage follower and a current amplifier (both realised using Operation Amplifiers (OpAmps)), as shown in 1.9. This ensures that current does not flow in the RE (as from requirements mentioned in subsection 1.1.2) due to the high input impedance of the Operational Amplifier (OpAmp). Additionally, the current amplifier allows to change

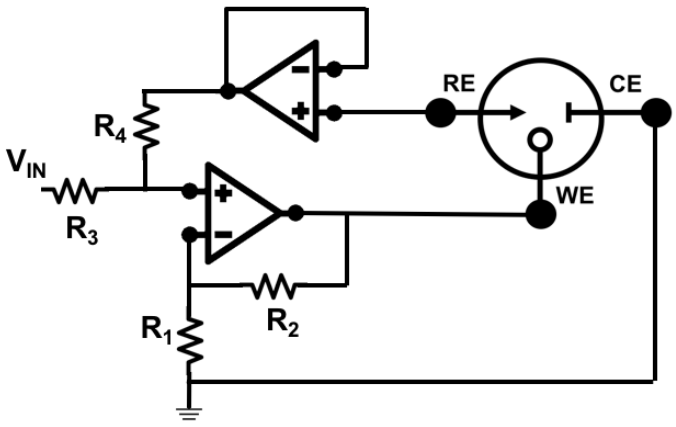


Figure 1.9: Grounded CE Configuration

its gain depending on the range of current flowing it the sensor, helping achieve better system

sensitivity. Only disadvantage of this configuration is the use of many discrete components. This makes the design prone to error due to mismatch in the manufacturing processes or make the circuit unreliable with temperature variations.

The second configuration, grounded working electrode, eliminates the need of many components. The WE is fixed to ground using a Transimpedance Amplifer (TIA), that amplifies the current flowing between CE and WE by its feedback resistance R_f and converts it to an output voltage V_{out}, as in 1.10. This use of the TIA ensures ground at the WE because of the virtual short property of the OpAmp, that holds as long as it operates in non-saturated regime. The same property is employed for applying the same voltage to RE and CE, and ensure the current flow only through CE and WE.

In both configurations, from a designer point-of-view, it is crucial to exploit the advantages of

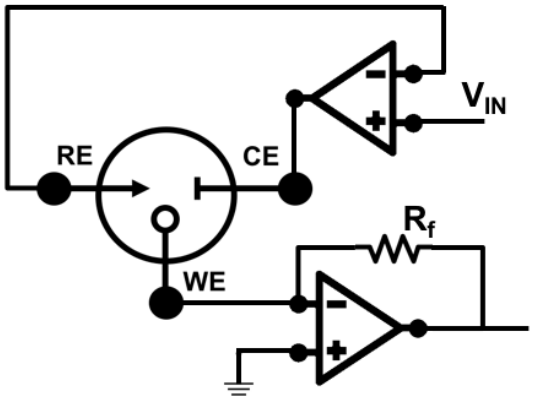


Figure 1.10: Grounded WE Configuration

OpAmps, e.g negligible input current, high input impedance, high voltage gain and low input offset, while making sure that it won't saturate while operating.

All this analog circuits that interfaces directly with the sensor (i.e. the electrochemical cell) are generally referred to as Analog Front-Ends (AFEs). Shown in 1.9 and 1.10 are only to possible designs, indeed both configurations can be developed using many different architectures.

When realising sensors instrumentation, an AFE is usually combined on-chip with additional circuitry to provide a more reliable capture of current information. This is done using *read-out* circuits, that convert the current from the analog to the digital domain using Analog-to-Digital Converters (ADCs) [2]. The same principle is also applied to the input path of the AFE, where

[2]Other techniques such as current-to-frequency conversion are also possible, if interested please refer to [19]

Digital-to-Analog Converters (DACs) are employed to create the biasing stimulus. A typical structure of AFE combined with biasing and read-out circuitry is shown in 1.11.

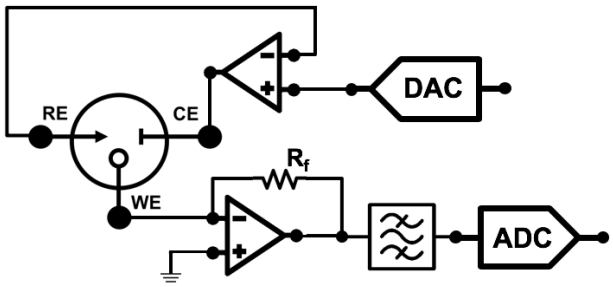


Figure 1.11: Grounded WE Configuration

1.4 Electrochemical Measurements

Depending on the type of bio-marker of interest different methods are used for detecting their concentration. Some of them rely on the ability of electronics to stimulate a reaction in the solution creating a current to detect (*amperometric detection*), others detect a variation in physical composition of the bio-electric interface (*capacitive or impedance detection*). The three types of measurements taken into account in this work are EIS, CA and CV. The first belongs to the category of capacitive or impedance measurements, while the others are two types of amperometric detection.

1.4.1 Electrochemical Impedance Spectroscopy

Similar to electrical impedance spectroscopy, EIS aims to scan in frequency the behaviour of an electrochemical medium, deriving the properties of its impedance. This, in bio-sensing, helps to find correlations between changes of electrochemical impedance and the presence of a target bio-marker in a sample. A countless number of applications make good use of this correlation in the area of medical diagnostics: e.g. bacteria detection [1], DNA detection [2], hormones monitoring [3], cancer markers detection [4], human proteins sensing [5].

In what follows the basic principles of impedance analysis will be discussed. Further, methods used for performing impedance measurement will be described.

Principles of Impedance Analysis

By definition, electrical impedance is defined as the opposition that a circuit presents to a current when a voltage is applied. More generally, impedance is expressed as a complex number obtained as the ratio between the complex AC voltage applied and the respective complex AC current flowing 1.6.

$$Z = \frac{|V|cos(\omega t)}{|I|cos(\omega t - \theta)} = \frac{|V|e^{j\omega t}}{|I|e^{j[\omega t - \theta]}} = R + jX \tag{1.6}$$

The real part of the impedance is defined as *resistance* (R) and the imaginary part as *reactance* (X). In practice, the relation is often represented in its polar form and showed on the complex plane (as shown in figure 1.12). In this way, real and imaginary parts are related to amplitude

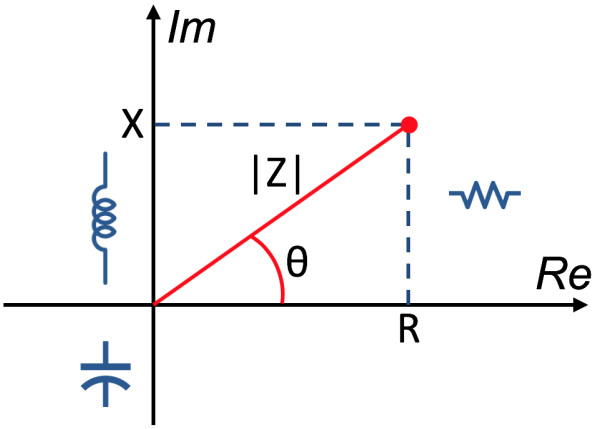


Figure 1.12: Complex representation of the impedance vector

($|Z|$) and phase (θ) of the complex vector representing Z (equation 1.7).

$$\begin{cases} |Z| = \sqrt{R^2 + X^2} \\ \theta = tan^{-1}(\frac{X}{R}) \end{cases} \tag{1.7}$$

$|Z|$ is related to the difference between current and voltage amplitude, and θ to the phase shift between the two. The sign of X (i.e. the sign of θ) determines the entity of the reactive component: capacitance if negative, inductance if positive. Therefore, when current *lags* the

voltage X is a capacitance, and when current *leads* the voltage X is an inductance. On the other hand, the more the current is attenuated with respect to the voltage the smaller the resistance is.

It is important to highlight that this definition of impedance is valid in the case of linear time invariant (LTI) systems, exhibiting three conditions: (i) linearity, (ii) stability, and (iii) causality. Indeed, if this definition wants to be applied, this conditions would need to be assured also when treating non-electrical impedance.

For analysing an impedance, one method consists in applying a sinusoidal voltage to the device under test while measuring the resulting current, so called "potentiostatic" measurement. By extracting the amplitude difference and the phase shift between voltage and current impedance parameters can be evaluated. Similarly, the same method can be used applying a current and measuring a voltage, in a "galvanostatic" measurement.

When this technique is used at multiple frequencies, impedance spectral behaviour can be evaluated. In this case, we refer to the measurement as *Impedance Spectroscopy (IS)*. Results of this type of analysis are generally shown in form of Bode or Nyquist plots. The first are used to show the spectrum of an impedance's amplitude and phase, the second to plot the change of its real and imaginary parts with frequency.

Other than for examining electrical impedances, IS can be also used for other mediums, like electrochemical solutions, therefore becoming a genuine bio-sensing technique. This application has been called *Electrochemical Impedance Spectroscopy (EIS)*, and consists in using IS for stimulating electrochemical cells. Although, electrochemical impedance and electrical impedance are not the same. Precautions have to be taken for obtaining a linear time invariant system (i.e. using small signals for operating in pseudo-linear region)[12]. Moreover, biasing conditions have to be assured (as discussed in subsection 1.1.2) therefore the sole potentiostatic method can be utilised.

EIS is performed in three steps: (i) stimulus of the device, or sample, under test, (ii) acquisition of the resulting current output, and (iii) processing of the value of impedance (figure 1.13).

These three tasks can be accomplished using various techniques that will be discussed in detail hereafter.

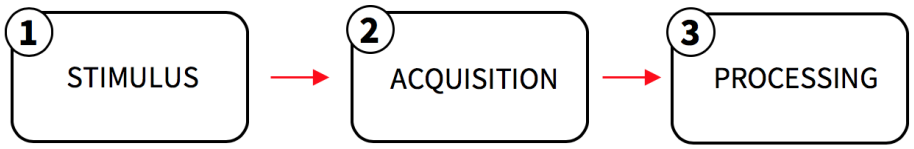


Figure 1.13: Block diagram of an impedance measurement

Stimulus Generation Techniques

The types of stimulus used for EIS can be summarised in three categories: (i) *single-sine*, (ii) *multi-sine* or *wavelet*, and (iii) *white noise*. These are respectively assocciated with three generations of EIS: frequency response analyzer (FRA), first-generation Fourier transform EIS (1-FTEIS) and second-generation Fourier transform EIS (2-FTEIS) [20].

Single-sine consists in applying successively small AC waves in pulses of increasing frequency, whereas multi-sine combines more frequencies in a single signal (creating a wavelet). This second methods requires a different type of processing involving fast Fourier transformation (FFT). Last, applying the same concept, white noise can be applied to the cell for obtaining the spectral response of the system after elaborating the response with FFT.

For creating sinusoidal stimuli Direct Digital Synthesis (DDS) is generally used. This technique produces an analog sine wave by reading digital values stored in a Look-Up Table (LUT) in memory and sending them to a Digital-to-Analog Converter (DAC). A phase counter (called "phase accumulator") increments its value continuously, triggered by a clock signal. The value of phase is then converted into amplitude by looking at the respective value of the LUT. The working principle of DDS is shown in figure 1.14.

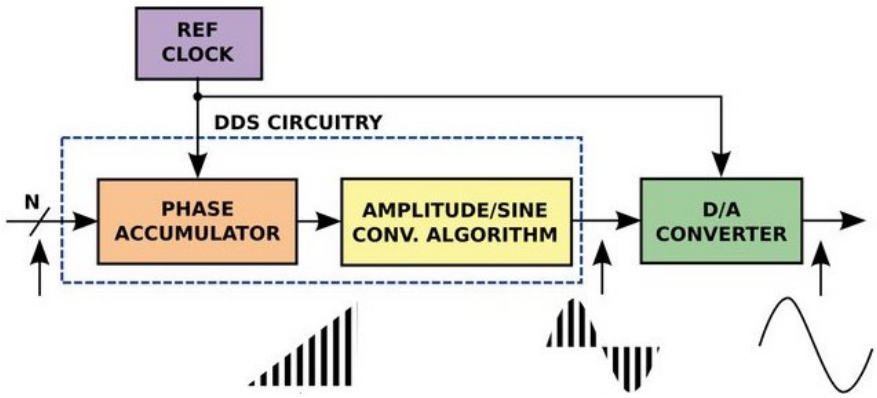


Figure 1.14: Block diagram showing the principle of DDS

Acquisition and Processing Methods

Acquiring the output current, and converting it into the digital domain for processing, involves read-out and sampling techniques. Since most ADCs have voltage inputs, current from the cell is always converted into voltage by a TIA before the analog-to-digital conversion. Filters are often put after the TIA for restricting the signal bandwidth to avoid aliasing. Depending on the processing method one or two ADCs are used (with fixed or variable sampling rate).
Finally, for processing acquired samples and obtaining the value of impedance, three methods are used: (i) *I-Q method*, (ii) *FFT*, (iii) *sine fitting*. The I-Q method uses extra hardware to extract from the output signal the in-phase and quadrature components. Measuring these two the amplitude and phase of the signal can be direcly extracted [21]. Second, fast fourier trasformation (FFT) can be used for extracting the impedance value from the deconvoluted signal. Last, sine fitting algorithms can be applied to the gathered samples to obtain amplitude and phase shift of the output of the system with respect to the applied input [22].

1.4.2 Chronoamperometry

Other than measuring the impedance of the electrochemical cell, in this work we are interested in measuring the presence of analytes correlating it with the RedOx current generated at the Bio/CMOS interface. As previously described, the relationship between current increase and molecular concentration is given by Cotrell's equation (equation 1.4)[19]. In chronoamperometry (CA) a fixed voltage is applied to the electrochemical cell for generating a change in current by triggering the RedOx reaction of the analyte. The higher the concentration of bio-marker in the solution, the higher would be the output current.
For testing sensors instrumentation, CA is usually performed while continuously injecting doses of analyte, obtaining a response like in figure 1.16. Usually current is also plot versus the concentration of analyte for obtaining *calibration curves*. From this curves, parameters characterising the bio-sensing platform can be extracted (e.g. the *sensitivity* or the *limit of detection (LOD)*).

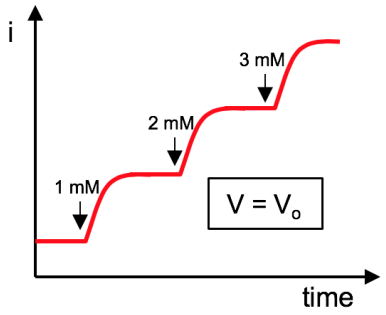


Figure 1.15: Example of chronoamperometry graph

1.4.3 Cyclic Voltammetry

Another electroanalytical method for sensing bio-markers is cyclic voltammetry (CV). This consists in the measurement of the output current of an electrochemical cell while applying a voltage varying cyclically.

CV is used to evaluate, in the current response, the presence of peaks related to reduction or oxidation of analytes. This enables not only to evaluate the concentration of a compound (by measuring the height of the peaks), but also to sense multiple analytes reducing or oxidising at different voltages. An example of CV is shown in figure 3.21. The position of the peaks for a

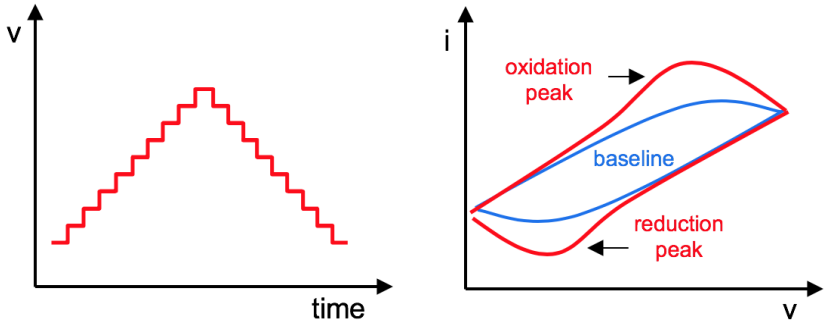


Figure 1.16: Example of CV voltage stimulus (left) and result (right)

specific analyte is given by the equilibrium between oxidation and reduction. This is evaluated using Nerst equation [19]:

$$E = E^0 + \frac{RT}{nF} ln\left[\frac{C_O(0,t)}{C_R(0,t)}\right] \tag{1.8}$$

where E is the external applied potential, E_0 is the standard potential, R the universal gas constant, T the temperature, n the number of electrons involved in the semi-reaction, C_O the concentration of oxidised metabolite, and C_R the concentration of reduced metabolite.

1.5 Portable Impedance Analysis Devices

Since the aim of the project is designing a portable device for electrochemical measurements and impedance spectroscopy, the state-of-the-art of current devices performing this kind of measurements has been reviewed. This has been focused mainly on portable devices for IS to be possibly adapted for EIS, CA, and CV using additional hardware.

A first approach to portable IS found in literature uses microcontroller based architectures coupled with various front-ends [23, 24]. These studies show good results in IS and bioelectrical impedance analysis (BIA), with large impedance and frequency ranges. Although, both this works do not focus on the application of the device to electrochemical cells. Also, even though they describe fairly small devices, any assessment has been made about their usage as portable devices.

A more complete result has been obtained by another study using a more powerful ARM microcontroller in a portable architecture [25]. This uses solely a μC and general purpose operational and instrumentation amplifiers. Also, it uses DDS for producing the stimulus and sine fitting algorithms for processing results. The obtained result has great features, but it is not suitable for interfacing with bio-sensors and it is not optimised for power consumption.

Other studies assess the possibility of creating implantable devices for impedance measurements. These show very small and low-power devices for EIS [26] and for bio-impedance analysis [27]. These two approaches are certainly very promising for future applications of EIS in portable, wearable and implantable devices, but do not show great measurement features. More importantly, due to the complexity and the time needed for the development of such devices, they have not been taken into account for eventual implementation in this work.

Regarding implementations on PCB, many studies in literature make use of the component AD5933, by Analog Devices®. Among these, three have been taken as reference for their implementation of the AD5933 in: a portable impedance analyser [28], a wearable impedance analyser [29], and a bio-sensing platform for protein and DNA sensing [30]. This device is indeed of very easy implementation and has shown satisfactory results in many applications. For this reason it has been considered in this work for possible use in the final PCB.

More recently, a newer integrated alternative from Analog Devices® , the ADuCM350, has shown interesting results in PCB-based EIS applications for cortisol sensing [3] and for nitrate sensing [31]. Due to its very low power nature, and its features as impedance analyser, the ADuCM350 has also been evaluated for possible implementation in this work.

1.6 Chapter Summary

The basic principles of bio-sensing have been presented. Key concepts have been described regarding: electrochemical mediums, equivalent electrical circuits, hardware topologies for bio-sensing, and the three types of electrochemical measurements taken into account in this work. Finally, a brief summary of literature regarding EIS devices has been reported.

Chapter 2

Energy Harvesting in Portable Devices

The ability of powering mobile electronic devices using ambient energy or energy produced by the the human body itself is nowadays, from an engineering point-of-view, one of the most fascinating challenges. Energy scavenging would give portable electronics the ability of self-powering themselves, extending their lifetime and easing the burden on battery requirements [32]. Many ways of producing electricity from renewable sources on large scale have been explored in the past. Only recently, the emergence of low-energy electronics highlighted the importance of miniaturising those sources for providing, locally, small amounts of power. This gave birth to the field of energy harvesting.

In the area of bio-medical sensors, researcher tried in the past to use energy scavengers (e.g. self-winding wristwatch generators for powering pacemakers [33]), yet not achieving very remarkable results. Indeed, the large gap between bio-sensors' power requirements and the power generated by small energy scavengers is an impediment. Despite this, recent developments of new materials and techniques for harvesting human power, together with the optimisation of low-power electronics, give hope that this gap could be shortly filled.

This chapter will focus on what types of energies are suitable for harvesting in portable and wearable devices and what are the methods for empowering them with energy harvesters.

2.1 Harvestable Energy

For mobile applications, the critical trade-off in the use of energy harvesters is between size and generated power. In this work, the sources which have been object of study are limited to the ones that can be physically suitable for portable and wearable applications (i.e. small enough to be carried without impacting on ones everyday activity).

For finding the type of energy the device would witness the most, and therefore the one more suitable to be harvested, the placing of the device for its final use has to be considered. Indeed, available human power varies largely depending on the placement or the harnessed activity [34]. For a portable device we can assume that one would carry it in a pocket, while a wearable device needs to be placed on a specific part of the body.

This section focuses on ways to gain energy from the act of carrying around a device and touching it while using it. Four types of sources of energy have been selected: (i) pressure, (ii) frictional force, (iii) applied motion and (iv) heat.

For transducing this forces into energy, physical properties of materials are exploited. In particular: *piezoelectricity*, for generating energy from pressure, *triboelectricity*, from friction, *magnetic induction* from the movement of a magnet, and *thermoelectricity*, to tranduce heat into electric voltage. Two crucial parameters that have to be taken into account in the choice of harvesting source are: (i) conversion efficiency of the transduction, and (ii) harnessed electrical quantity (voltage or current).

2.1.1 Piezoelectric

Piezoelectricity is the propriety of some materials of generating electric potential difference when subject to mechanical stress, and, on the opposite, of producing elastic deformation when stimulated by an applied voltage. Thanks to this quality, piezoelectric materials (e.g. quartz, ceramics, etc.) find use in many applications, both as sensors and as actuators.

In their use in energy harvesting, piezoelectric transducers are employed either as diaphragms or as cantilevers (shown in figure 2.1). This structures enable to create a high voltage (tens of

Figure 2.1: Commercial piezoelectric transducers

volts per cm^2) by pressing or bending the ceramic material. This happens due to the creation of a surface charge density caused by a change in the polarization vector (P) of the material. This change can be obtained by changing: the orientation of P, the symmetry of the crystal, the applied mechanical stress. Piezoelectric transducers are sometimes used in a stack and connected in parallel for increasing their current output. In terms of efficiency of piezoelectric transduction values found in literature range from 20% to 40% depending on the materials and the structures used [34, 35].

2.1.2 Triboelectric

Another way of creating voltage gathering charges on a material's surface is using triboelectric effect. This happens when certain materials come into frictional contact with a different material becoming electrically charged. This phenomenon is common in everyday life when experiencing the buildup of static electricity, and the consequent electrostatic discharge (ESD). Physically, triboelectric effect is due to the exchange of charges that happens when two surfaces are put in contact with each other, caused by an imbalance in their electrochemical potential. When detached, the two materials obtain a net charge imbalance between each other, therefore electric voltage is generated. The generation of charges is enhanced when rubbing two materials together because of the repeated contact and separation of the two surfaces.

The efficiency of triboelectric transducers can reach values of 40% and 50% [35]. In addition, the materials used for fabricating triboelectric generators are usually light and can be flexible or elastic. In practice, using triboelectric effect for energy harvesting requires the creation of

custom generators, since there are no commercially available products to fulfil this function.

2.1.3 Magnetic

Electromagnetic or magnetic induction is the physical phenomenon used for creating a flow of current in an electrical conductor by changing the magnetic field around it. This is mathematically described by Ampere's circuital law, which relates the integrated magnetic field around a closed conductor to the electric current passing through it.
In energy harvesting, magnetic induction is utilised to translate the movement of a magnetic object into electrical current, by letting it slide within a unit with winded coils. An example of application of this method is found in shake-powered flashlights (in figure 2.2). The intrinsic

Figure 2.2: Example of magnetic harvesting: shake-powered flashlight

efficiency of magnetic induction is much higher than the other presented method, although it reaches high values only in case of continuous and periodic movement [36]. Moreover, no commercial magnetic generators are available. Indeed, magnetic generators are usually made with custom coils and different types of commercially available magnets.

2.1.4 Thermoelectric

Thermoelectric effect is used for converting temperature differences into an electric voltage. This is achieved using components called *thermocouples* (in figure 2.3). The effect is reversible, so it is also possible to generate a temperature difference by applying a voltage. Thermocouples are commercially available, light and inexpensive. Although, efficiency of thermoelectric

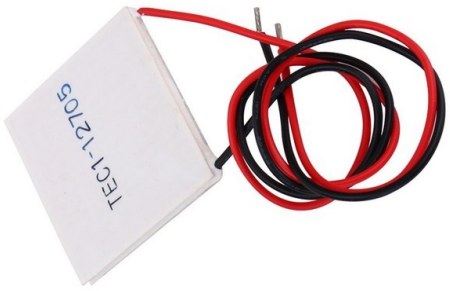


Figure 2.3: Example of thermocouple: Peltier thermoelectric cooler

generators is very low (around 10%)[37]. This means that they can be used either in case of a very large and constant applied temperature difference, or for large surface areas.

2.2 Harvesting Techniques

An energy harvesting system usually consists of three components: (i) an energy generator, (ii) a conditioning circuit and (iii) a storage device [38] (figure 2.4).

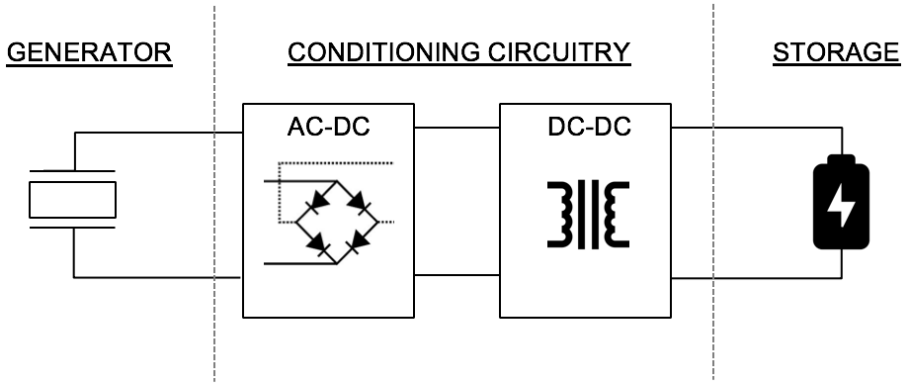


Figure 2.4: Energy harvesting block diagram

It has been previously shown what types of energy generators could be employed in portable devices for harvesting energy. Nevertheless, the electrical quantity generated by this sources has to be treated by proper conditioning circuitry in order to be utilised for powering a device. This circuit topologies take a current or voltage input and generate a stable current source for charging a storage device. Since the quantity generated by these types of transducer varies in time, a rectifier is usually employed as input of the conditioning circuitry. Then the voltage is

fed to DC-DC converters used for generating the supply voltage or charging the storage. This circuits are usually either implemented with discrete components, or integrated on-chip in components called "energy harvesters". For implementations on PCB, due to the complexity of the topologies for achieving highly efficient conversions, the second approach is undoubtedly the most convenient.

When energy is harvested, it has to be stored either in batteries (e.g. lithium-ion batteries) or in capacitive elements (conventional capacitors or supercapacitors). The choice between this two types of storage determines the use of the energy harvesting module: either as battery support, or as direct source of supply. The main trade-off that as to be considered for choosing between the two types of devices is between energy density and power density (as shown in figure 2.5). In general, other than for their energy delivery capabilities, the main advantage in

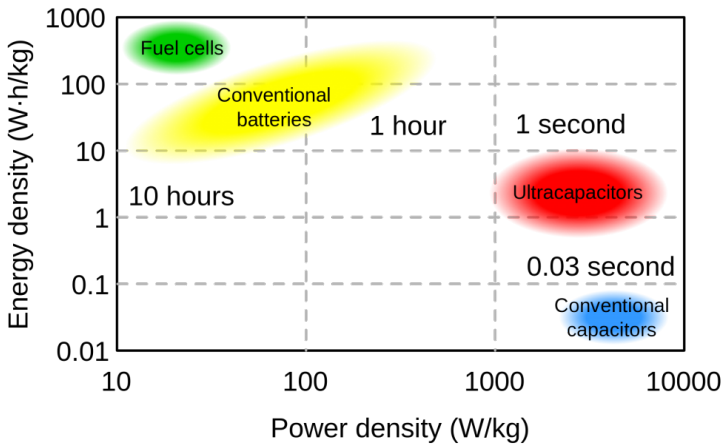


Figure 2.5: Trade-off between energy and power density in energy storage devices

the use of supercapacitors with respect to conventional batteries is in their lifetime. Indeed, charged is stored in supercapacitors electrostatically, without involving any chemical reaction. On the other hand, batteries store energy electrochemically, and their charging and discharging can degrade their capabilities over time.

2.3 Portable Energy Harvesting Devices

For the aim of this project, a research has been done in literature to determine the most suitable harvesting configuration to be implemented. Of the multiplicity of articles found in literature describing the use of harvesting sources, the ones that had an impact on the design choices of the device developed will be described. Due to physical constraints and commercial availability of components for the eventual implementation the analysis focused on: piezoelectric, triboelectric and magnetic harvesting.

Regarding the use of piezoelectric transducers, more works try to exploit their ability of generating charges by placing them under a shoe to gain energy from movement, either with pressure transducers [39, 40] or with complex moving hydraulic systems [41]. These are able to gain considerable amounts of powers from heel strike, from 10mW with piezoelectric soles up to 250-700mW with hydraulic piezoelectric actuators. Another interesting study makes use of commercially available piezoelectric diaphragms in different configurations, achieving an electrical power generation of up to 28mW [42].

Piezoelectric harvesting, other than used by itself, is often combined with magnetic harvesting for increasing the efficiency of the system, in particular when devices are subject to both pressure and vibration/movement. A study [43] shows how the use of such hybrid structures can increase the output power with respect to piezoelectric-only or magnetic-only harvesters up to 118%. Another [44], proposes a use of commercially available conditioning circuitry for realising hybrid harvesting systems. A comparison between the two sources alone is also presented in [45], concluding that magnetic harvester are far more efficient than piezoelectric, although they may be uncomfortable in some applications due to their higher mass and movable parts.

Indeed, magnetic harvesting has the potential of allowing high efficiency energy transduction by human motion, when certain design conditions are met [46]. Efforts have been made in order to move from linear systems, like shake-powered flashlights, to three-dimensional structures [47]. This, with a 3 cm diameter semi-sphere shaped device, is able to provide up mW of power by vibration in transport applications.

Another very promising technique for harvesting human motion is shown in [48]. This exploits

the same idea used for in-sole piezoelectric harvesters, but using triboelectric transducers. A light layered structure is able to produce, by simply palm tapping the transducer, a continuous electricity source of around 1mW.
Finally, also triboelectric harvesters have been combined with magnetic generators. A good example in wearable devices is shown in [49]. This study proposes an hybrid energy harvesting structure used for powering a wrist-watch with a generated power up to 8mW.

2.3.1 Considerations and Objectives

As a result of literature review and considerations based on theoretical limits of different harvesting technologies and approaches, it has been decided to develop an hybrid harvesting system for powering the final PCB device. This, to be possibly utilised with multiple sources, e.g. piezoeletric, triboelectric, magnetic and thermoelectric, singularly or combined. To achieve this, the idea is to use of commercial energy harvesters in a novel configuration, in combination with a supercapacitor to realise a battery-less device. In this way, for applications such as portable personal diagnostics, the device could have a continuous source of power generated by the human body. This would allow a slow charge of the energy storage for later use in energy consuming tasks like bio-sensing and impedance analysis measurements.

2.4 Chapter Summary

A concise description of possible harvestable forces in portable devices has been presented, with focus of what kind of transducers are suitable for each application. Techniques for generating power from these sources have been briefly presented and the main trade-offs in the choice of an energy storage for portable application have been cited. Last, a summary of recent literature studies concerning energy harvesting methods for use in portable devices has been presented.

Chapter 3

Prototype for Bio-Sensing and Impedance Analysis

At this stage, it has been assessed the possibility of obtaining CA, CV and EIS measurements using a system composed of: (i) an AFE for three-electrode electrochemical measurements and (ii) a microcontroller unit, as shown in figure 3.1. This, not only for evaluating the feasibility of the system itself, but also for understanding the main challenges and limitations that have to be faced in the design of a device of this kind.

For this part of the project the workload has been divided in four sections: (i) embedded software development, (ii) setup characterisation, (iii) data processing software development, and (iv) final testing and measurement.

First, in the following, the devices used for the realisation of the system are presented. Secondly, an outlook of the system is shown, with a description of each block's function. Finally, the results of the setup testing are reported, focusing both on the obtained measurements and on the limitations and trade-offs to be taken into account in the ensuing stages of the project.

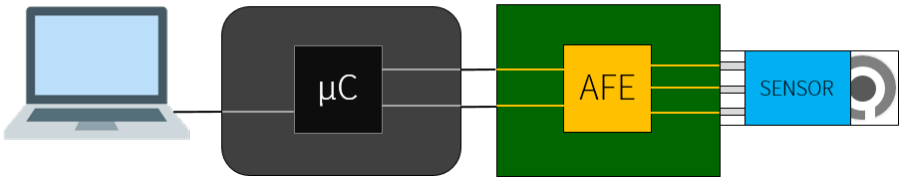


Figure 3.1: Envisioned setup for protoype #1

3.1 Components choice

Custom Multipurpose AFE: Mooncake

As previously discussed in chapter 1, to perform an electrochemical measurement in potentiostatic mode an AFE is needed to interface with the sensor. For this purpose, a multipurpose AFE developed at the Centre for Bio-inspired Technology (CBIT), known as Mooncake, has been used.

Mooncake is an integrated bio-sensing platform consisting of four AFEs and four readout circuits for electrochemical measurements. It features on-chip four three-electrode cells (Pt electrodes), pH sensing units (ISFETs), and heaters for temperature control. It also provides, for each one of the cells, a 10-bit DAC for stimulus and a 10-bit Analog-to-Digital Converter (ADC), which can be controlled externally using a Serial Peripheral Interface (SPI) bus. Moreover, peripherals and outputs can be selected for use or bypassed using a programming word through SPI.

The key unit of the chip is the Differential Readout IC (DiRIC) [50], which consists of both a potentiostat and a readout circuit. DiRIC uses a switch-capacitor architecture that synchronises the sampling of the sensors' current and the ADC, hence enhancing noise immunity of the measurement removing background current in the analog domain. Further, thanks to the nature of the circuit, the gain of the current-to-voltage conversion is a function of system clock frequency, therefore obtaining a tuneable current dynamic range (0.47pA-20μA [50]). Indeed, this latter feature made Mooncake a great candidate to be used in this application. In 3.2 a top level block diagram of Mooncake's architecture for a single sensor. For the purpose of this project, only one of the four sensor interfaces has been used.

For connecting Mooncake to a μC the chip needs to be bonded to a package and placed on

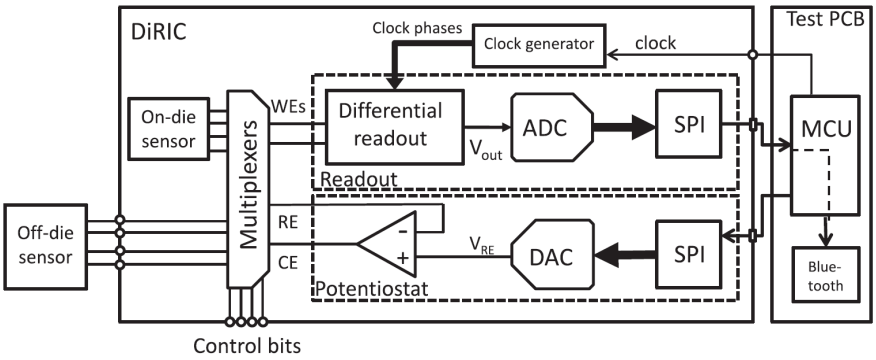


Figure 3.2: Top level diagram of Mooncake's single sensor's architecture

a testing PCB. As a matter of fact, the connections on the chip are micro-pads, almost not visible to the naked eye. For testing, these are connected to an 84-pin JLCC package by means of gold micro-wires. After bonding, the device can be then placed in a socket on a PCB containing pins and conditioning circuitry. All these conditioning steps have been done prior to the beginning of this project, right after Mooncake's fabrication. The testing PCB, shown in 3.3, has been designed in coordination with the development of the embedded software, so that the μC outputs could match the respective chip input pins at best.

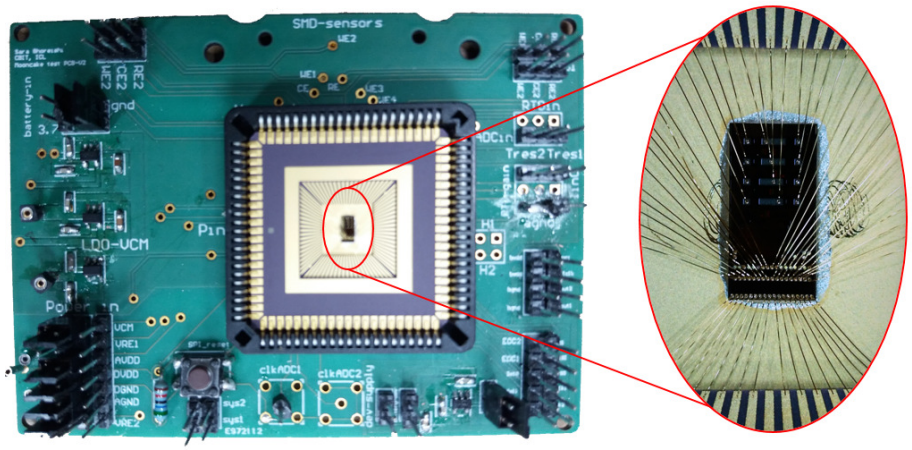


Figure 3.3: Testing PCB with socket for hosting Mooncake, with input pins and conditioning circuitry

Microcontroller

By its own, Mooncake cannot perform a full electrochemical measurement, and needs to be interfaced with a device for processing sampled data, for providing the system clock, and for programming is tuning registers. This can be done using a μC able to communicate to the chip through SPI and with the ability of providing a clock source.
Due to its availability, a development board from Freescale®, FRDM-KL26Z [51], has been considered first for this purpose. Featuring a Kinetis® μC with a 32-bit ARM Cortex-M0+ core, with clock frequency up to 48 MHZ, 128KB flash, 16KB RAM, and multiple interfaces (as shown in 3.4), amongst which two 16-bit SPI and three Timer/PWM modules, the board satisfied all the initial requirements and has been selected for interfacing with Mooncake. The decision of using a development board, shown in 3.5, other than implementing the μC device as single component on a prototype board, has been taken for simplifying the implementation and speeding up the prototyping stage. Indeed, a development board contains all necessary conditioning components for the device functioning, including a debug interface to connect directly the μC to a PC through a USB port for programming and debugging. Also, it can be programmed immediately using the Integrated Development Environment (IDE) provided by the manufacturer and an open source toolchain (GNU). Thus, all firmware has been developed in C language using "CodeWarrior®" and available libraries for the μC in use.

Processing Unit

Last, a unit for data post-processing and as user interface is needed. For simplicity's sake, a Graphical User Interface (GUI) on Matlab® has been developed to act both as a controller for setting up the measurement parameters and for post-processing and showing the results to the user. A serial communication has established between PC and μC configuring two General Purpose Input/Output (GPIO) pins as Universal Asynchronous Receiver-Transmitter (UART) input and output, and connecting them using a USB-to-UART cable to the PC.

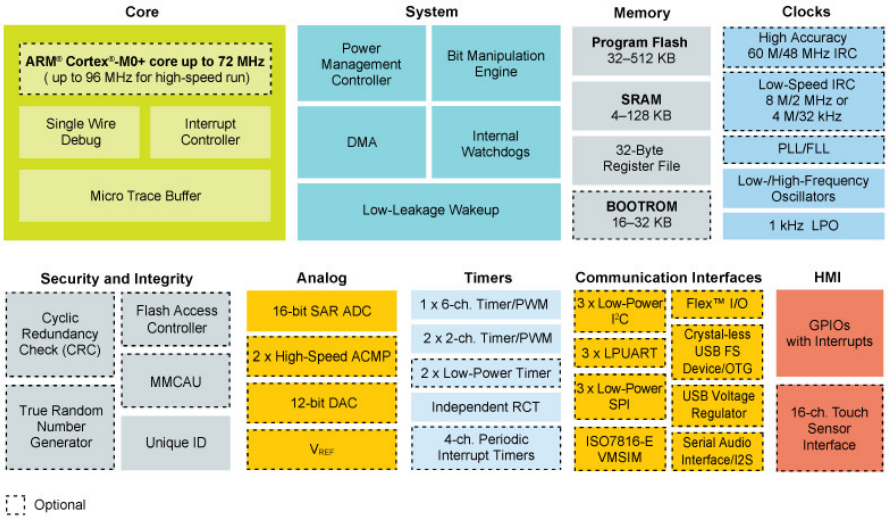


Figure 3.4: Kinetis® μC L Series KL2x Block Summary

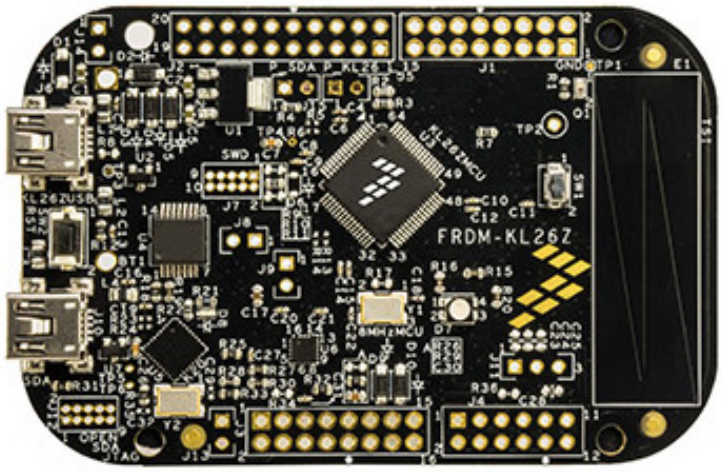


Figure 3.5: Freescale® FRDM-KL26Z Development Board

3.2 Setup Overview

As shown in the block diagram in figure 3.6, the setup is composed of two boards, the FRDM-KL26Z and the PCB for interfacing with Mooncake. The configuration contains all the required hardware blocks for electrochemical measurements and EIS, i.e. stimulus, biasing, and readout (subsection 1.4.1), and for communication to a user console on a PC.

Here a description of the involved blocks, ordered following the stimulus signal path:

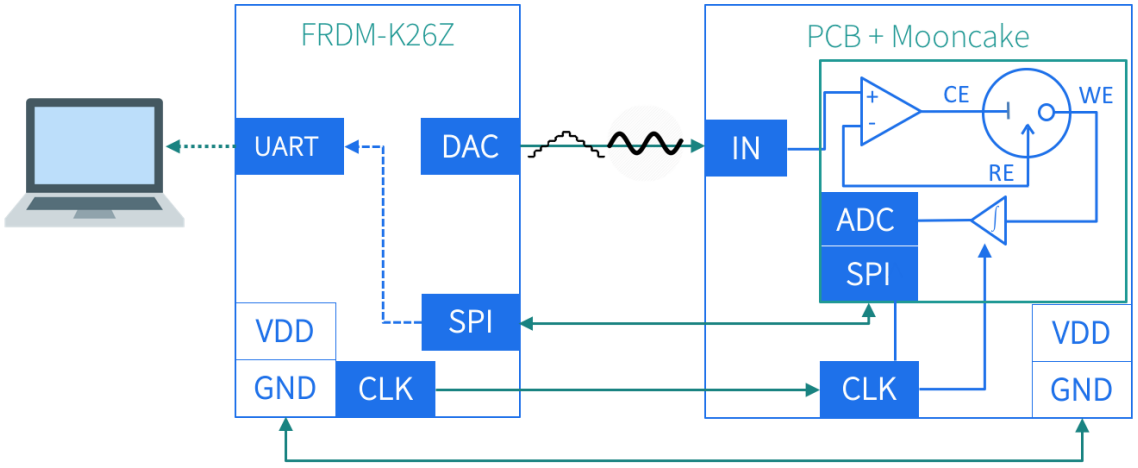


Figure 3.6: Prototype #1 top level system block diagram

DAC and DDS

The voltage generation for biasing the electrochemical cell has been achieved using the 12-bit DAC on the μC, adopting a DDS technique. The DAC on the μC has been used instead of the one on Mooncake in order to obtain a better amplitude resolution, i.e. a smaller Least Significant Bit (LSB), crucial parameter for meeting the requirement of minimum stimulus amplitude for EIS.

According to the type of measurement, the stimulus has been synthesised using a different software routine and different peripherals. Indeed, producing a sinusoidal signal has more limitations with respect to a staircase, mainly due to a higher amount of data to be moved from memory to the DAC output register. A flow chart describing the software routine for stimulus generation is shown in 3.7. When performing CA, since voltage has to be constant for the whole measure, the value of the DAC register has been set only once at the beginning of the routine. For CV, a staircase shaped signal has been created at the output decreasing and increasing successively a fixed "step" value from the initial voltage of zero volts, until performing a whole cycle. Zero volts has been chosen in order not to stimulate the cell (i.e. the solution) during the pause between one cycle and the next one. Finally, for EIS, stimulus generation requires an extra step for calculating the amplitude values for each point of the sinusoidal signal and creating the LUT (i.e. the phase accumulator for DDS, as described in subsection 1.4.1). Furthermore, for accurately setting the time between two DAC register updates, crucial property

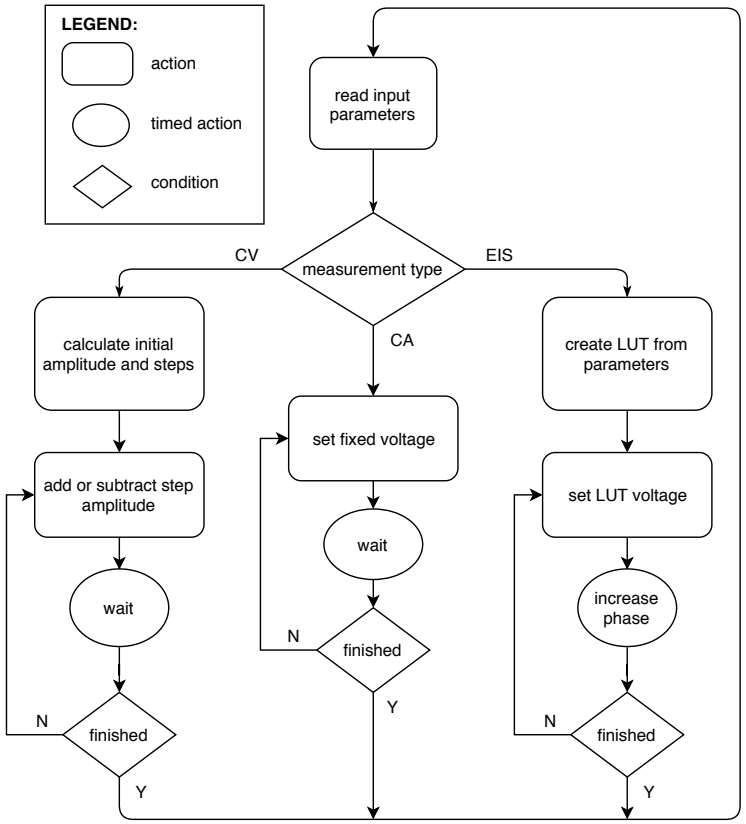


Figure 3.7: Flow chart representing the software routine for stimulus generation

for tuning precisely the output frequency of the sinusoidal stimulus, one of the μC's timers has been used. This has been set to generate an interrupt at each time step. Therefore, a faster or slower output sine (higher or lower frequency) has been obtained by reading the LUT with a faster or slower rate, calculated in units of timer ticks (time for one cycle of the timer's clock). On the other hand, in CV, the time between two DAC updates has been set using wait states in the routine.

Finally, during EIS, the stimulus frequency has been swept for covering the selected frequency spectrum. This has been achieved pausing the output stimulus at its offset value at end of each frequency, changing the value of timer ticks according to the next frequency, and resuming the measurement for the next step.

AFE and Read-out

Mooncake takes care of biasing the electrochemical cell in order to stimulate the electrodes and read the output current. Peripherals and outputs can be selected using 16-bit programming word sent through SPI.

The available configurable options are shown in the following table:

Mooncake Peripherals Selection & Programming Word					
Name	RE1 Source	Transducer	# Cells	WE Output	WE Selection
Tuning bit	15	14	13	12	11
Option(bit)	Pins (1) DAC (0)	Electrodes (1) ISFETs (0)	Single (1) Pin (0)	ADC (1) Double (0)	#1 (1) #2 (0)

The AFE circuit is designed in grounded-working configuration. A voltage follower at the out-

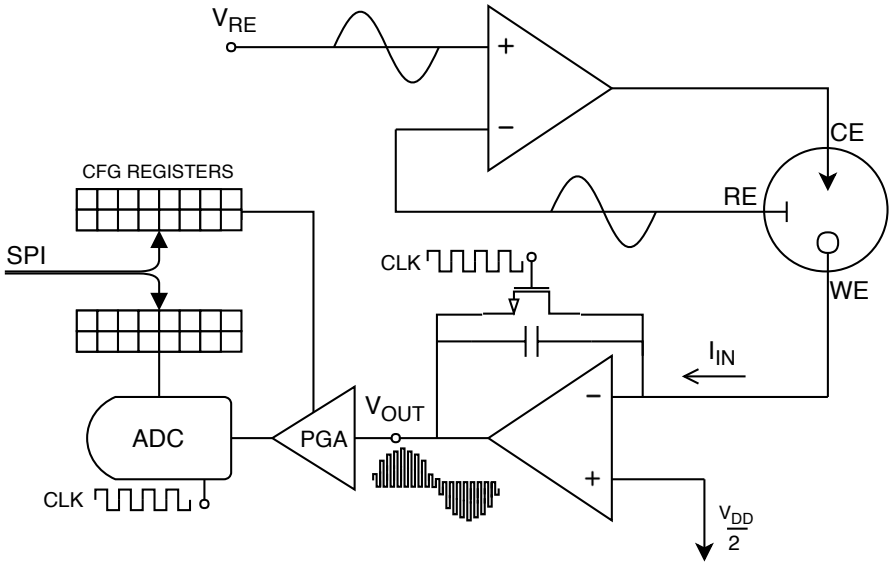


Figure 3.8: AFE functional block diagram

put makes sure the current flows in the CE and a stable voltage is applied at the RE. A switched capacitor TIA at the input picks up the current flowing in the WE and converts it into a voltage, while applying at the WE a fixed voltage equal to half the supply voltage. This so that, changing the RE voltage, the cell can be biased both with a direct or reverse voltage. Furthermore, for adding an extra voltage gain to the readout, a Programmable Gain Amplifier (PGA) is placed between the TIA and the ADC input, with a tunable gain (range 1-16) set using an

additional SPI word. A graphical representation of the AFE configuration is shown in 3.8.

As mentioned, a relevant peculiarity of the input stage is the ability of changing the gain of the current-to-voltage conversion with the clock frequency. This property is given by the nature of the conversion, actuated using the timed charge and discharge of the TIA feedback capacitor, by means of a transmission gate. A slower clock gives in fact more time for the capacitor to charge, i.e. for the voltage to rise, and therefore makes the stage "sensitive" to smaller currents. On the opposite, discharging the capacitance more often helps avoiding the amplifier's saturation and a larger dynamic range for the input signal can be obtained. The current-to-voltage gain of the TIA stage is given by the relation $V_{out} = -\frac{1}{C_f}\int_0^{\frac{1}{f_{clk}}} I_{in}(t)dt$, that for constant current becomes $V_{out} = -I_{in}\frac{1}{f_{clk}*C_f}$, with V_{out} as the output voltage, I_{in} as input current, C_f feedback capacitance and f_{clk} clock frequency.

For choosing the right frequency for a measurement, i.e. the right gain, a rough estimate of the sensors current is needed. This can be done by characterising the sensor using a commercial potentiostat (i.e. laboratory instrumentation). Then, Mooncake's clock frequency is chosen according to the gain equation, so that the output voltage is large enough to be detected (a minimum peak voltage of some ADC LSBs - 1 LSB = ~3mV) and small enough not to exceed the output dynamic range (0.6-2.6V). An example of frequency choice is reported in 3.9, where an excel sheet embedding the gain equation is used for choosing between different frequency values.

	A	B	C	D	E	F
1	**Vdd (V)**	3.3				
2	**C (F)**	1.50E-11	**GAIN_PGA**	1	sysclk [Hz]	V_adc_max [V] (< 2.6 V)
3	**V_cm (V)**	1.65			100	1.02E+02
4	**I_we (A)**	3.00E-07			200	5.17E+01
5					400	2.67E+01
6					800	1.42E+01
7					1000	1.17E+01
8					2000	6.65E+00
9					4000	4.15E+00
10					8000	2.90E+00
11					10000	2.65E+00
12					20000	2.15E+00
13					40000	1.90E+00
14					80000	1.78E+00
15					100000	1.75E+00

Figure 3.9: Frequency choice excel sheet, with input parameters in green

Other than for setting up the TIA gain, the clock is also given to the ADC. When the on-chip

ADC is enabled, a clock divider block is triggered. This takes the system clock as input signal and outputs another clock 11 times slower to be used for shunting the integrator capacitance. Therefore, the ADC clock, 11 times faster[1], makes sure the the entire ADC conversion can take place between one "sampling" (i.e. shunt of the integrator feedback capacitance) and the other.

SPI

SPI communication is used between Mooncake and μC for two reasons: first for sending the programming words to the chip's registers, second to read the sampled values from the ADC.

Configuration Registers - Mooncake		
Cell	Address	Description
Middle	0x01	15-9 configuration bits, 8-0 ADC output
	0x02	13-10 PGA gain , 8-0 DAC setup
Top	0x05	15-9 configuration bits, 8-0 ADC output
	0x07	13-10 PGA gain , 8-0 DAC setup

Table 3.1: Configuration registers map and description

Mooncake's programming registers' structure is shown in 3.1. For setting up the chip's parameters one needs to sequentially select each of the registers and send a 16-bit SPI transfer. There are two registers for each couple of Mooncake's cells, called "middle" and "top". Each register can be chosen using a 2-bit multiplexer selection word, set using GPIO pins. In the code on the μC, programming words are composed according to measurement's parameters sent from the PC and then fed to a function that takes care of the sequential transfer.

On the other hand, for transferring ADC samples, the cell couple used for the measurement needs to be selected, then an SPI transfer can read from the register containing the sampled values. In this case, additional care has to be given in managing the timing of the transfer, in order to avoid reading the same sample twice. For this, the End Of Conversion (EOC) signal of the on-chip ADC is given as input to a GPIO pin on the development board, informing the μC when a new sample is available. Samples, as collected, are stored in an array indexed with the same value of phase used for the DAC's LUT. This enables, in the post-processing steps, to

[1]The value 11 is taken from ADC specifications and indicates the number of clock cycles needed for one conversion

know when the sample was taken with respect to the stimulus creation, providing information about the phase delay of the signal.

UART Communication

Last, samples have to be sent to the PC for post-processing. This is done by means of a serial transmission established between μC and Matlab®. The script running on the PC gathers all samples at the end of each measurement, post-processes them when needed, and shows the result in form of plots and graphs.

UART is also used for communicating to the μC the measurement parameters. These vary depending on the type of measurement. With the help of a simple GUI the user can communicate the parameters to the μC, start a measurement, and get the results. After selecting the type of measurement to perform, a dialog window for inserting parameters appears (figure 3.10) according to the method the user wants to use.

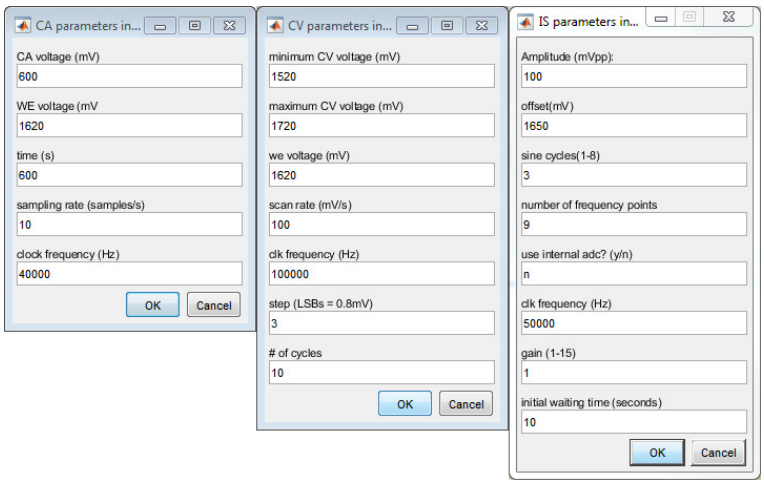


Figure 3.10: User interface for inserting measurements parameters

3.3 Setup testing

3.3.1 Stimulus tests

First, the limitations in the creation of a stimulus have been evaluated. This aspect is particularly critical in the creation of sinusoidal signals for EIS measurements. Indeed, compelling requirements of frequency range and stimulus amplitude must be met, and these are deeply influenced by the system's configuration and performance.
These tests have been performed connecting the output of the μC's DAC from the development board directly to a digital oscilloscope. Two stimulus parameters have been assessed: output frequency range and minimum amplitude. To do this, EIS sweeps have been performed and the output signal has been evaluated by reading the oscilloscope.
Furthermore, the functionality of the stimuli for all the three types of measurements has been verified.

Frequency range

Frequency is limited by two factors: the LUT length and phase resolution. A shorter LUT translates into less DAC updates, therefore higher achievable stimulus frequency at the same system clock. However, the LUT length also determines the amplitude resolution of the stimulus, and a too short LUT can result in a stair-shaped sine. After testing, a table summarising the achievable frequencies with respect to the LUT length has been compiled, shown in 3.2. The reason of this limitation will be further discussed in the following section. In configuring

LUT Length vs Stimulus Frequency	
256	∼1kHz
128	∼2kHz
64	∼4kHz
32	∼8kHz

Table 3.2: Limitation of stimulus frequency with respect to the LUT length

the setup, a rule of thumb has been adopted to avoid a stair-shaped stimulus. This consisted

in keeping the LUT length as long as 4 times the number of DAC LSBs (1 LSB = ∼0.8mv) necessary for representing the signal's peak (e.g. for a sinusoidal signal of 50mV peak amplitude: $\frac{50mV}{0.8mV}$=∼63, therefore a minimum length of 63*4=252).

Another relevant parameter for determining the LUT length is phase resolution. The LUT length represents in fact the number of units in which a circle representing the sinusoid phase is split. A "LUT step" can be converted in degrees to obtain the minimum detectable phase shift (i.e. phase resolution). Thus, for achieving a minimum resolution of 1 degree, the minimum LUT length is 360 steps. This because, as explained, the detection of phase shift between DAC stimulus and ADC samples is achieved by labelling received data with the LUT index, allowing the extraction of delay between signals in post-processing steps.

Minimum Amplitude

Regarding the minimum stimulus amplitude, this is theoretically limited by the DAC resolution, i.e. the dimension of the LSB step. A discrete sinusoidal signal has in fact deviates from a perfect sine wave due to quantization error, that generates quantization distortion. This phenomenon creates spurious frequency components in the stimulus signal. In this work, in which we are interested in a stimulus not smaller than 40mV peak-to-peak with a 12-bit DAC, quantization distortion is not a major limitation. In practice, since the voltage is not directly applied from the μC output to the sensor, the most limiting factor for minimum amplitude is external noise. Tests performed on noise, and consequent adaptations of the system, will be further discussed in the following section.

Stimulus Functioning

Last, the proper functioning of the stimulus signal had to be checked for each type of measurement. Results for CV and EIS are shown in 3.21, and 3.12.

As we can see in the CV example, which involves larger voltage excursion with respect to the EIS stimulus, the DAC output displays spikes at certain amplitude values. This is due to the

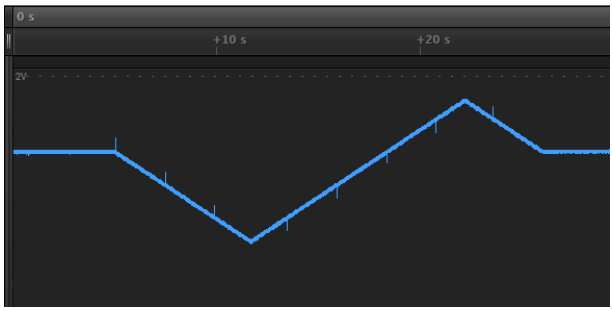


Figure 3.11: Example measured stimulus for CV measure

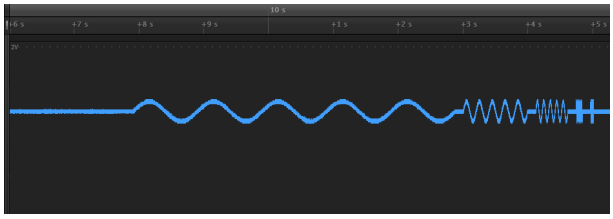


Figure 3.12: Example measured stimulus for EIS measure

the digital-to-analog conversion architecture, built using a so-called R-2R ladder. Indeed, an input voltage is converted into an analog value using the bits' values to trigger digital switches. Resistances are weighted so that the output voltage is proportional to the value represented by the respective bit. In case multiple bits change simultaneously, the delay between switches can cause a temporary higher current than expected, thus creates a spike at the output. Generally, the higher the number of changing bits the more significant the phenomenon is. For this reason, voltage transitions that change the most significant bit (i.e. cross the half supply voltage) should be avoided, in both CV and EIS stimuli.

3.3.2 Noise tests

For decreasing noise on the output signal tests with different setup configurations have been performed: first, to identify the most significant noise sources, second, to attenuate their contribution both on the stimulus signal and on the readout. Tests have been carried out with all connections between μC board and Mooncake's PCB in place. Noise has been evaluated at the RE pin of the PCB interface with the sensor, while a clock was provided to the chip.
The most relevant cause of noise has been identified in capacitive coupling between PCB tracks.

Indeed, an external digital clock connected to the testing PCB makes the current on its tracks go back an forth at every clock edge. This induces noise in all other tracks, and particularly in the ones parallel to the clock path. This phenomenon was found present both on the testing PCB and on the μC development board. Furthermore, faulty ground connections between PCB and μC board have been also identified as noise source. This is due to ground bouncing phenomenon caused by a too small resistance of the cables for the return current path.

A measurement of the RE stimulus (50mV amplitude, 2Hz frequency), generated by the μC and probed at the PCB's RE output pin, is shown in figure 3.13. The measure shows a roughly

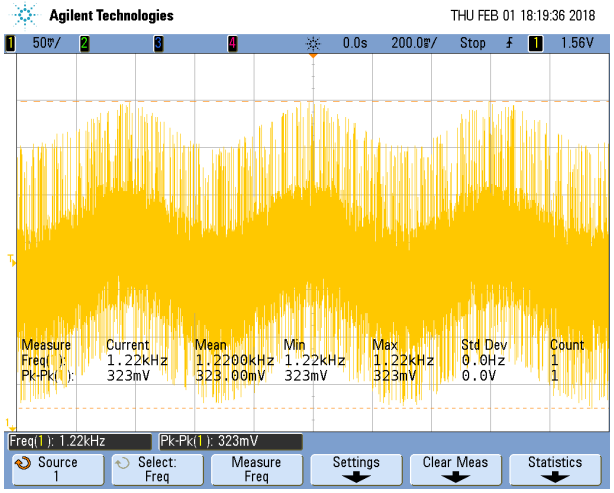


Figure 3.13: Noise measure before changes

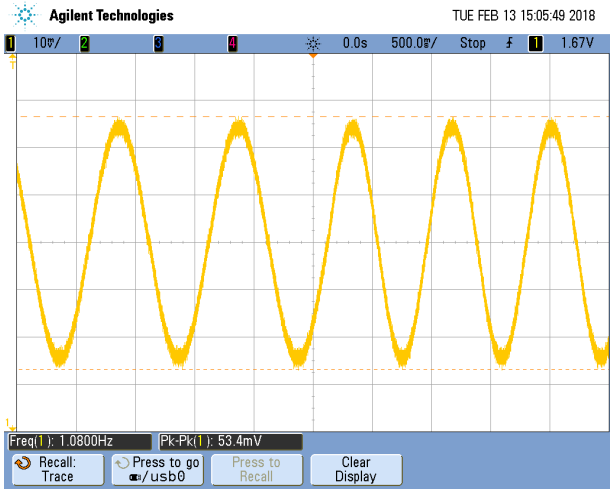


Figure 3.14: Noise measure after changes

300mV peak-to-peak noise superposed to the output, at the frequency of the provided clock (12kHz). The screen-shot in figure 3.13 has been captured before taking any action for noise reduction.

For attenuating noise, some changes in the setup have been made. On the μC board, GPIO pins for generating the clock and creating the stimulus have been kept away from each other, and all floating pins around the clock output have been internally connected to ground. Then, on the testing PCB, the track used for the clock was initially parallel to the one for the RE stimulus. Therefore, to reduce coupling, the track has been cut manually and an alternative path, away from the analog signal, has been used for providing the clock to chip. For grounding, two additional wires have been used for connecting the development board ground to the PCB, for a total of three connections. Finally, supplies have been kept separated, using a battery for the AFE and USB for the development board. The result of the changes is shown in figure

3.14, were the same pin as figure 3.13 has been probed. In this case the AFE has been stimulated with a 50mV peak-to-peak, 1 Hz sinusoidal signal. Noise contribution has been evidently successfully reduced.

3.3.3 Post-processing tests

As discussed in subsection 1.4.1, for determining EIS results a post-processing step is necessary in order to extract the amplitude and the phase of the signal with respect to the applied stimulus. For CA and CV post-processing can be also be useful for determining steps' values and peaks, but its use is not essential for determining the functionality of the system, therefore it has not been used.

Concerning EIS, two post-processing techniques have been adopted and tested: zero-crossing detection and sine fitting.

In a first assessment, in order to decrease the amount of data to be transferred through the serial port, a zero-crossing algorithm has been implemented on the μC. This, after gathering the samples for one frequency step, has been used to estimate the phase at which each sine cycle crosses its zero-amplitude value and then calculate an average phase between multiple cycles for each frequency point. Unfortunately, due to the noise present on the signal, identifying exactly the peak's phase, i.e. within the minimum phase accuracy, is trivial. In testing, results within ± 2 degrees from the ideal value have been obtained, but resulted also inconsistent between successive measurements. To achieve a more accurate result, a longer measurement with more cycles to average has been tried without achieving good results. The maximum number of measure cycles is in fact limited by the overall measurement time and cannot be increased enough to compensate the uncertainty caused by the noise.

In a second stage, a more noise immune amplitude and phase extraction step has been applied: sine fitting. For this, all data has been transferred to the PC, that run the fitting algorithm on Matlab®. This has been performed using three-parameters sine fitting (IEEE Std 1057), with as input the samples vector previously filtered with a digital zero-phase low-pass filter. This further technique has shown much better results than the previous, with a measured phase

variability smaller than 1 degree after offset compensation in the frequency range of interest. Figure 3.15 shows the result of the filtering and fitting step and figure 3.16 the change of phase error compensation in the bode diagram of a 1MΩ resistor.

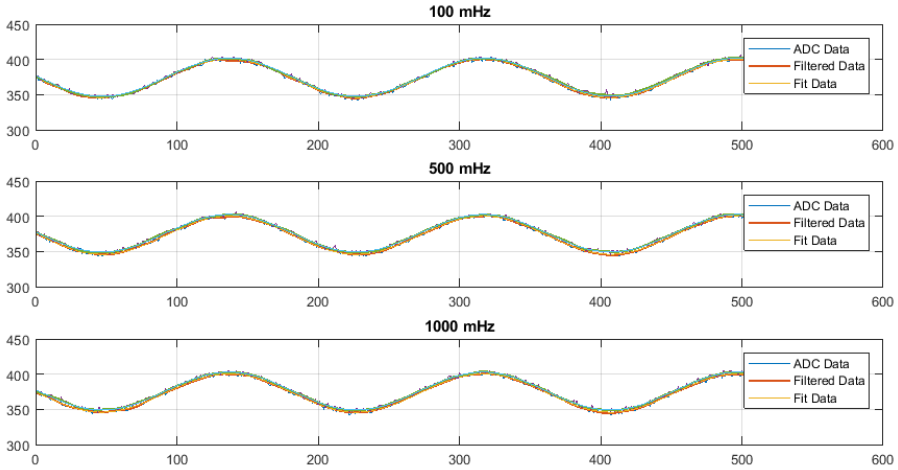


Figure 3.15: Raw output current sinusoidal data with superposed filtering and sine-fitting results

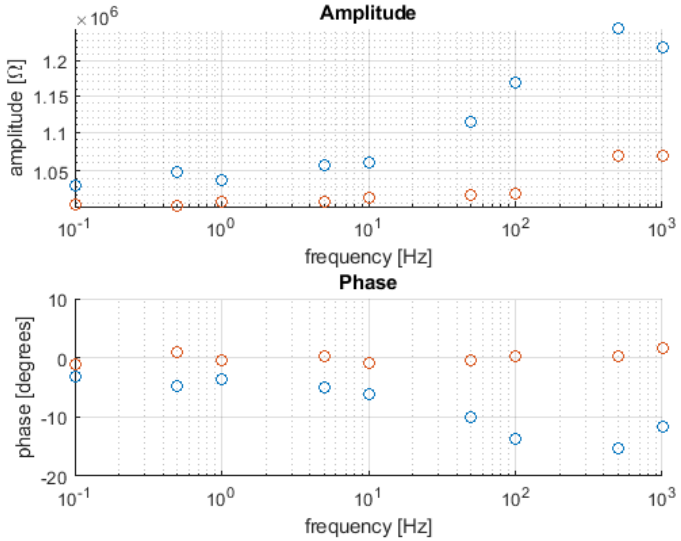


Figure 3.16: Bode diagram for a 1MΩ resistor: blue and red markers for before and after compensation respectively

3.3.4 Measurements

Finally, all three types of measurement have been tested. Experiments have been conducted using Phosphate Buffer Saline (PBS) solution (pH: 7.4, 100mM) as sample solution and hy-

drogen peroxide (H_2O_2) as target compound. Changes in concentration have been achieved by injecting successively 2μL of H_2O_2 (diluted with deionized water in a stock solution at 0.1M) in 20mL PBS solution. A Dropsens®DRP-550 Screen Printed Electrode (SPE) has been adopted as electrochemical cell for the measurement. It hosts on its surface a WE (4 mm diameter) and a CE made of platinum (Pt), and a RE made of silver (Ag). For ensuring a stable and dry setup wires have been soldered to the sensor for connecting to the AFE pins and all contacts have been covered with dry epoxy. The whole setup is shown in figures 3.17 and 3.18.

Figure 3.17: Prototype setup for electrochemical measurements

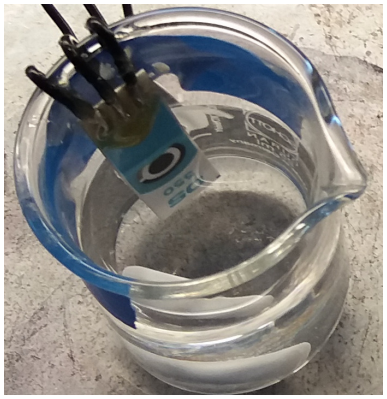


Figure 3.18: Detail of sensors contacts covered with dry epoxy

As reference, all measurements have been previously executed using laboratory instrumentation (CH instruments® 600E Electrochemical Workstation).

Chronoamperometry

CA has been performed by applying a 600mV voltage stimulus at the WE and measuring the cell amperometric response. In order to establish the gain of the AFE according to the expected

current range of the sensor, measurements with laboratory instrumentation have been priorly performed (figure 3.19).

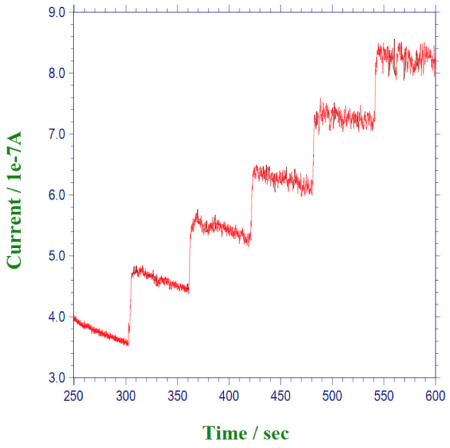


Figure 3.19: Chronoamperometry 0μM to 500μM H_2O_2 with laboratory instrumentation

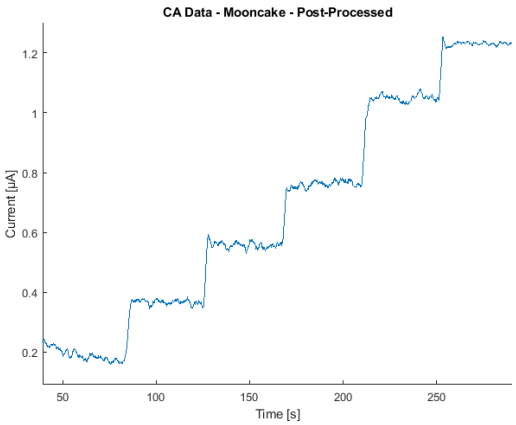


Figure 3.20: Chronoamperometry 0μM to 500μM H_2O_2 with prototype setup

Before the first injection a time of around 1 minute has been used for stabilising the output current. Then, one dose of 100μM H_2O_2 every minute has been injected for a total of five injections. The whole measurement has been performed in stirring condition using a hot-plate and a magnetic stirrer.

The same CA has been then repeated with the prototype setup, following exactly the same steps and using the same chemicals. The measurement result is shown in 3.20. The figure has been produced from the Matlab® script after post-processing steps and averaging. A good result has been obtained, with a current characteristic in the expected range.

Cyclic Voltammetry

Next, CV has been performed, again first with laboratory instrumentation and then with the prototype platform. The same change in concentration as CA has been used, even though CV revealed a much lower sensitivity than CA. Unlike CA, CV cannot be performed continuously in stirring condition, and the voltage stimulus needs to be paused at every cycle between one injection and the other. Since H_2O_2 oxidation peak is expected to be around +650mV, CV has been set for sweeping the voltage between -300mV and +700mV, with a scan rate of 100mV/s. Before the first injection multiple consecutive cycles need to be applied in order to obtain a stable baseline. When the current characteristic starts superposing the one of the previous cycle for multiple times a baseline is obtained and the measurement can start. Five consecutive doses of 100μM have been injected in the break between one cycle and the other, and stirred for around 20 seconds. The measurement result for laboratory instrumentation and prototype setup are show in figure 3.21. Unfortunately, system's constraints did not allow to get a good response from the prototype setup. The absence of a clear reduction peak was probably due to the way the voltage stimulus has been applied to the cell, different from the triangular shape used by the commercial potentiostat. Also to obtain the whole CV figure without saturating the AFE a small gain had to be used. This decreased the sensitivity of the system in detecting changes of the oxidation peak.

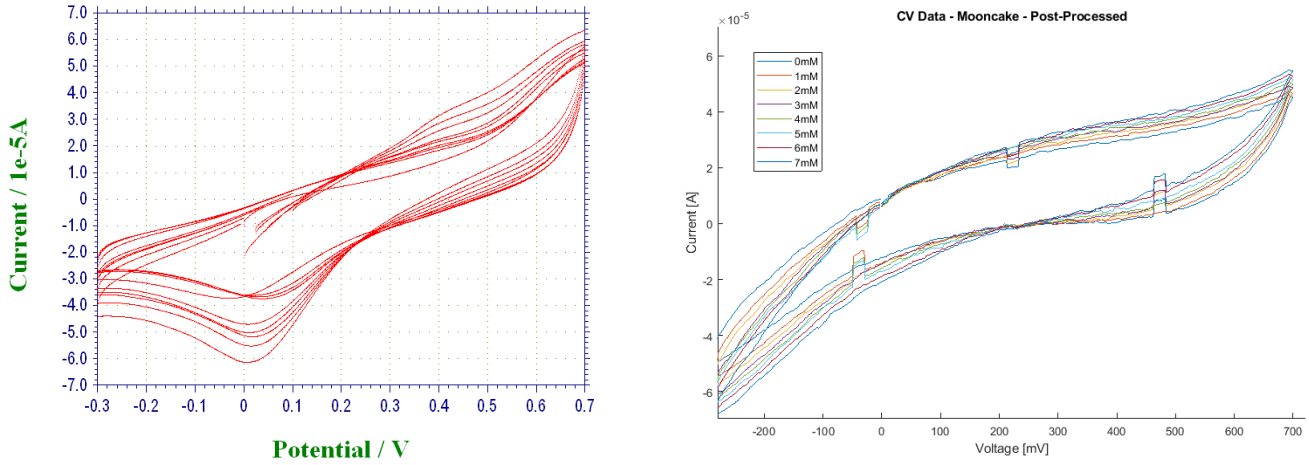


Figure 3.21: Cyclic voltammetry 0μM to 500μM H_2O_2: left laboratory instrumentation, right prototype setup

Electrochemical Impedance Spectroscopy

Last, EIS has been performed on a pure PBS sample with the same sensor and the same setup configuration. Due to system's limitations the spectral range has been limited to one decade (100mHz - 1Hz) and ten frequency steps. Before using the sensor, a calibration step has been performed. This consisted in executing one impedance spectroscopy, at frequencies of interest, with a know resistor in place of the sensor (as shown in 3.22). The phase offset of the calibration measurement can be than considered as offset of the whole system and used for compensation. A 660kΩ ±1% resistor has been chosen to stay in the same current range as the sensor and have an accurate value of impedance.

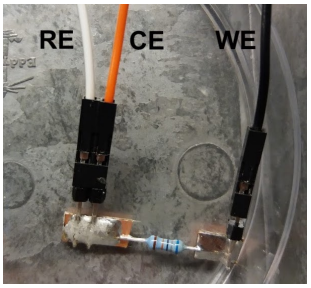


Figure 3.22: Calibration resistor connected to the AFE testing PCB

Results of the PBS measurement are shown in 3.23. As comparison, in 3.24 the same measurement performed with laboratory equipment. A satisfactory result has been reached. Further

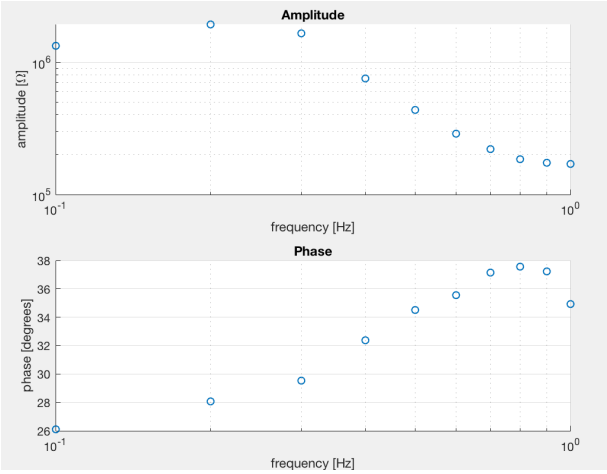


Figure 3.23: EIS of PBS solution performed with prototype setup

test could have been conducted for seeing change in impedance, reproducing the EIS behaviour

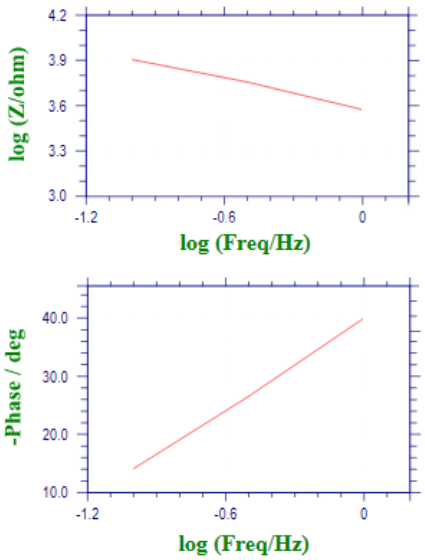


Figure 3.24: EIS of PBS solution performed with laboratory equipment

in sensing applications. Due to the many limitations of the system, that would have confined this analysis only at very low frequencies, it has been decided not to proceed with more test and move to the second next step of the project.

3.3.5 System's Limitations and Trade-offs

Storage Usage

Since post-processing has to be done externally, samples have to be stored on-board during the measurement and before being sent to the PC, not to let the slowness of the UART transfer be of impediment for the stimulus-sampling routine. This approach involves, for long measurements or high sampling rates, extensive use of memory.

Two options are available on the μC for storing data: the 16kbytes RAM or the 128kbytes Flash Memory. In a μC architecture, RAM offers higher transfer speed, i.e. with frequency comparable to the core's clock, while flash is much slower (generally more than 10 times), due to a lower clock frequency and a slower protocol for accessing memory cells. For this reason two approaches are used for storing data, depending on the type of measurement. During CA and CV, which involve lower sampling rates and less samples with respect to EIS, data is read

from the SPI and stored in the heap section of the memory till the end of a measurement. Whereas, for EIS, data is stored in the heap during each frequency step and then sent to the flash between one step and the other. This approach allows not to limit the measurement speed. Indeed, the use of an interrupt for scheduling DAC updates creates the need of performing all other operations, i.e. sampling the output, transferring the sample through SPI and saving it in the μC RAM, between two consecutive updates. Using directly the flash for saving the results would lead to longer memory access time, therefore longer time between DAC updates and slower stimulus. Indeed, it has been described how the delay between two DAC register updates determines the frequency of the output signal. However, if the speed of the updates is too fast, and all operation in the time frame between two updates do not have time to be concluded, the system's functionality is compromised.
The number of samples for consecutive measurements is therefore limited to a value smaller than 8000 (16kbytes RAM divided by a 2 bytes *uint16_t* sample), not considering all other variables in RAM. This block of memory is dynamically allocated at the beginning of every routine depending on the measurement parameters. The user is notified in case the chunk to be allocated exceeds the size of available memory.

Clock and Gain Limitations

As described, a clock has been provided to the chip for the functioning of its read-out block in sampling and converting current values. Also, we have seen that the provided clock (*CLKADC1*) is proportional to the gain of the TIA (lower frequency, higher gain). Therefore, the sampling rate of the read-out stage is linked to the gain of the current-to-voltage conversion. This, thanks to the clock divider block managing the synchronisation between TIA and ADC, should not generate any limitation. Although, unfortunately, it has not been possible to use the divider block, probably due to an issue in the implementation of the circuit on-chip. Therefore, the same clock has been used both for the ADC and for driving the transmission gate shunting the TIA's feedback capacitance, generating a trade-off between gain and sampling rate.
Moreover, another clock has been provided to the system, the SPI clock (*SCLK*). *SCLK*'s frequency has to be high enough to make sure the value is read from the on-chip ADC output

register in time (before it is overwritten by the next sample). Indeed, the SPI protocol is able to transfer 1 bit per clock cycle. Since the ADC output register stores a 16 bit value, for reading one sample 16 *SCLK* cycles are needed. Thus, in this implementation, *SCLK* needs to be 16 times faster than *CLKADC1*.

In testing, this relationships has lead to a limitation when trying to obtain a lower gain (increasing *CLKADC1*) for high sensor's current. Indeed, the clock generated by the μC has a limited output frequency that cannot exceed values of around 2MHz.

Peripherals Parallelism

In the use of a single core μC true parallelism cannot be achieved unless extra hardware peripherals are available for specific scopes. As mentioned, DDS has been used for generating the DAC stimulus. A timer has been employed for updating the output register with a LUT value at every interrupt. The time between two consecutive updates determined the frequency of the output signal. This method commits the core to working non-stop for producing the stimulus, and all other operations need to be performed in the time-frame between two updates. Using additional peripherals, for instance a Direct Memory Access (DMA) block connected between memory and DAC, can relax this limitation and create parallelism. Unfortunately, due to limitations in the use of the DMA module of the KL26Z μC, it could not be used for this scope. Therefore, extra care has to be taken for ensuring synchronicity (e.g. using instructions at lower abstraction levels). In a different implementation, this issue could be easily solved in hardware by using a DDS block on-chip.

3.4 Chapter Summary

A prototype setup for performing CA, CV and EIS has been assembled using a μC development board and a custom AFE for electrochemical measurements. Firmware for the μC has been developed for performing the three measurements and a small GUI has been built on Matlab® for interacting with the platform from a PC. The configuration has been tested and tuned, for

avoiding initial issues, by modifying interface connections and pins. Finally, the system has been validated performing all three measurements using a commercial sensor and varying H_2O_2 concentrations in PBS solution. System's limitations have been evaluated and the trade-offs to be taken into account in the design of a device of this kind have been discussed.

Chapter 4

PCB Conception, Realisation and Testing

The final step of the project concerned the conception, realisation and validation of a portable energy harvested PCB that could perform bio-sensing measurements and, in particular, impedance analysis.

The workload has been divided in three main sections: (i) conception and design of the PCB, (ii) development of embedded firmware and assembly of the board, (iii) tests and measurements. This steps have been scheduled with the help of a GANTT chart (figure 4.1)

In this chapter, first the outline of the system will be presented, showing a high level block

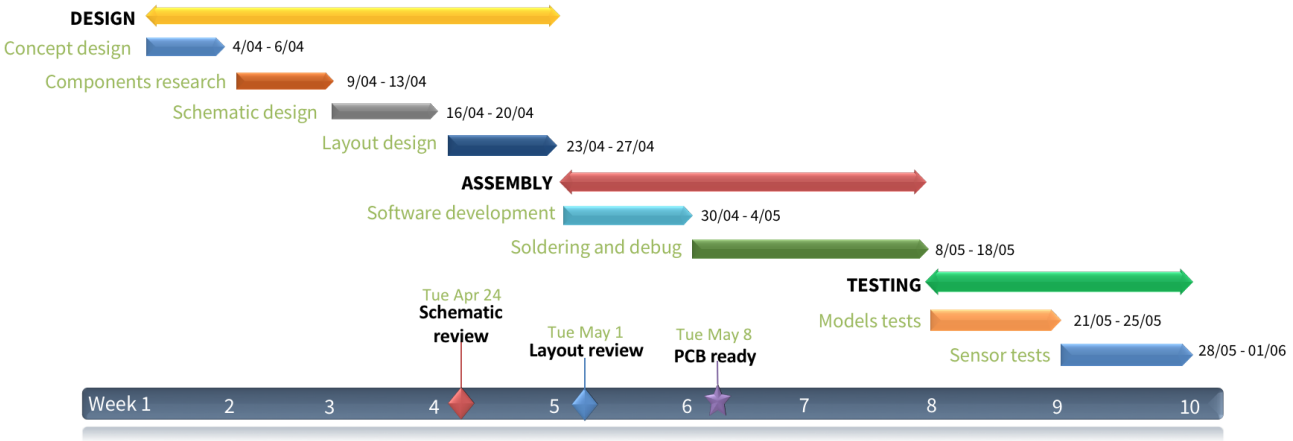


Figure 4.1: GANTT chart presenting the schedule for the realisation of the PCB

diagram of the system. Then, everything that has been done in the three scheduled stages will be described, focusing first on design choices and components selection, then on the realisation of concept designs, schematic and layout, and finally on the testing and verification of the assembled device.

4.1 System Outline

The system we aim to design is composed of: (1) a μC for managing measurements and elaborate the results, (2) an AFE for interfacing with an electrochemical cell, (3) a low-power communication block for transferring results to an external device, (4) supply circuitry with energy harvesting capabilities. This structure has been used as reference for conceiving the concept designs.

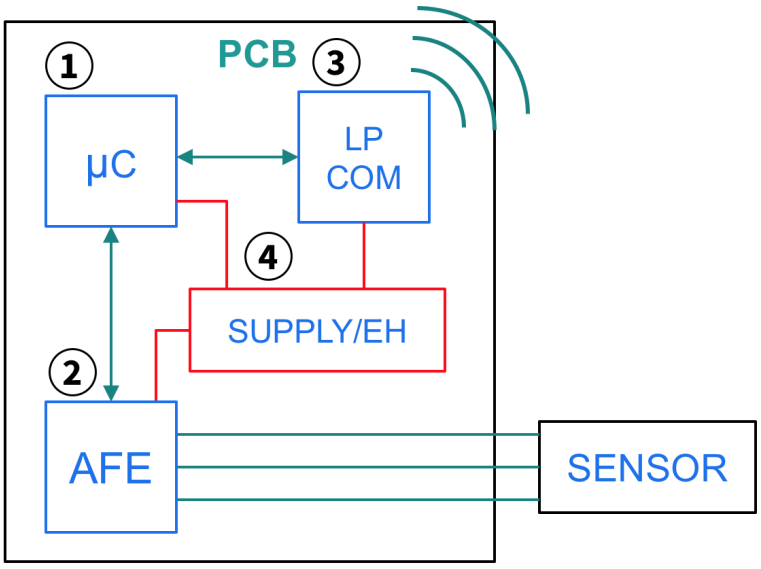


Figure 4.2: PCB high level block diagram

Before designing, the system requirements have been defined and summarised. First, at a higher level, in terms of what overall qualities the final device should have. Secondly, more in detail, regarding specific functions of the envisioned PCB (measurement, communication and supply). Specifications have been based on possible uses of the device and values have been either taken from literature or calculated from theoretical limits. All specifications have been reported in the following tables.

Device Requirements
• Portability (easy to carry in a pocket)
• Versatility (adaptable to various sensors)
• Low Consumption (sustainable by energy harvesting)
• Wireless Connectivity
• Bio-sensing Functionality (could perform amperometric and impedance measurements)

Performance Specifications	
Measurement	
Stimulus Amplitude	10 mV - 1V
Current Range	1 nA - 1mA
Frequency range	100mHz - 1MHz
Impedance range	100Ω - 100MΩ
Phase resolution	Changes < 1 Degree
Results processing	On-board
Communication	
Interface	Standard (SPI, I^2C)
Data-tranfer rate	> 32kbps (1K 32-bit sps)
Current consumption	mA range
Supply	
Provided current	mA range
Source	Harvesting + opt. battery

All the device features have been correlated with the respective required hardware:

- *Measurement* $\longrightarrow$ μC, DAC, AFE, ADC, Impedance Analyser
- *Communication* $\longrightarrow$ Low-power Communication Module
- *Supply* $\longrightarrow$ Energy Harvesters, DC-DC Converters, Regulators

Then, components research has been conducted on manufactures' websites and components distributors (*digikey.co.uk*, *mouser.co.uk*, *uk.rs-online.com*, *uk.farnell.com*). Different options of components suitable for this application, and available for buying, have been evaluated.

4.1.1 Microcontroller and AFE

As previously seen, for performing an electrochemical measurement the cell need to be stimulated with a signal generated by a DAC. Concurrently, an ADC is used to sample the output of the cell. This synchronisation generates limitations in the use of the two, when using a single-core μC. This issue can be solved with proper hardware choices to achieve parallelism between

the two blocks. For instance, two options could be: using a multi-core μC, or employing specific hardware to take care of one on the tasks autonomously (e.g. DDS on hardware with DMA). On the other hand, since most of the obstacles are related to impedance spectroscopy (due to a more onerous hardware usage), another option could be utilising specific hardware for impedance analysis (so-called impedance analyzers). This approach has been frequently seen in literature, in particular with the AD5933 (by Analog Devices®), and has been considered the best approach also for this work. A newer and interesting device is the ADuCM350, also by the same company. Together with a memory mapped impedance analyser, the ADuCM350 hosts a μC and many peripherals for communication and debugging. The components choice for this PCB block has been focused on this two latter devices.

AD5933 vs ADuCM350

The AD5933 is a device able to interface with an impedance, perform impedance spectroscopy, and send measurement results to an external μC using I^2C. It contains a DDS core, a read-out circuit and a post-processing unit (Discrete Fourier Transformation (DFT)) (figure 4.3). Thus, it can be programmed through I^2C and perform a measurement autonomously.

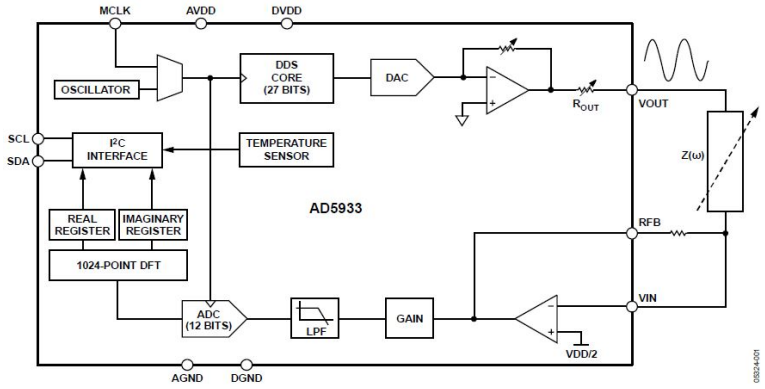


Figure 4.3: Block diagram of the AD5933

The device has both advantages and disadvantages. A summary is proposed in table 4.1.

As seen in chapter 1, many devices in literature exploit the advantages of the AD5933 and overcome some of its limitations. Overall, it has been considered as good candidate for this work for its capability, with proper hardware adjustments, to roughly cover all specifications.

Advantages	Disadvantages
DDS and read-out on-chip	Peak consumption (∼15mA)
Post-processing on-chip	External OpAmps for AFE
Accuracy (phase err $< 1°$)	Additional AFE for CA, CV
I^2C interface	Additional DAC for CA, CV
Impedance up to 100M	Additional ADC for CA, CV
Scalable stimulus to mV	
Low sleep current ($\sim 0.7\mu A$)	

Table 4.1: Summary of AD5933 pros and cons

Although, two critical weaknesses have to be taken into account: its peak current consumption, and the need of using plenty of extra components for its implementation. Indeed, the AD5933 needs an external μC for functioning and also an extra AFE for amperometric measurements. On the other hand, the ADuCM350 features a completely different structure (shown in figure 4.4). It embeds in a μC architecture a specific block for impedance analysis and electrochemical measurements (*"AFE CONTROLLER"* in figure 4.4). Also, it contains an AFE configurable for two, three or four electrodes measurements, many standard peripherals (GPIO, SPI, I^2C, etc.), and a debugging interface (JTAG, SWD).

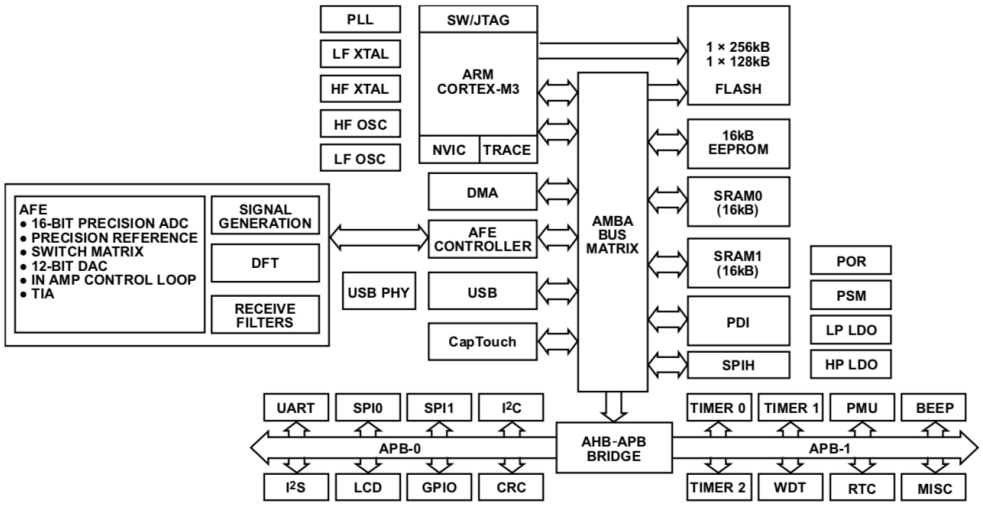


Figure 4.4: Functional diagram of the ADuCM350

This device, with the only addition of some conditioning components (bypass capacitances, calibration resistors and impedance blocks for setting the TIA gain), can be used to perform CA, CV and EIS interfacing directly to a sensor. Additionally, it is designed for low power consumption, using power gating and clock gating strategies, and provides various low-power modes. Moreover, the presence of an AFE on chip eliminates the need of designing an external

one, easing its implementation.

Finally, a comparison of some key parameters of the two options in terms of impedance measurement has been made. A summary is shown in table 4.2.

Overall, the ADuCM350 has been considered the best alternative for this implementation.

	AD5933	**ADuCM350**
Max current consumption	15mA	8mA
Stimulus amplitude	5mV - 3V	300μV - 1.2V
Impedance range	External TIA	External R_{TIA}
Frequency range	1mHz - 100kHz	230mHz - 75kHz
Processing	1024-Point DFT	2048-Point DFT
Additional circuitry	AFE, μC	None

Table 4.2: Comparison between AD5933 and ADuCM350 features

Although, because of its limitations in terms of frequency range and spectroscopy resolution (ADuCM350's DDS stimuls can vary in 0.23Hz steps only), initially it has been decided to conceive two different implementations. The first, with both ADuCM350, used as μC and for CA and CV, and AD5933, for compensating in impedance analysis for high and very low frequencies. The second, with only the ADuCM350.

4.1.2 Wireless Communication

Another quality of the envisioned PCB is the ability of sending results of a measurement to a portable device. For this, the most commonly used wireless connectivity methods in portable devices have been taken into account: *NFC*, *Bluetooth* and *Wi-Fi*.

Given the need of minimising power consumption and, at the same time, harvest energy from external sources, NFC has been chosen in this work for wireless communication. Indeed, some NFC devices on the market are able not only to use remote powering for the on-chip circuitry managing the communication, but also to harvest energy for external devices. This feature is crucial in the design of a battery-less device.

Nevertheless, with respect to Bluetooth and Wi-Fi, NFC bitrate is slower (hundreds of kbits/s vs Mbits/s) and communication range is shorter (centimeters vs meters). Although, NFC specifications are enough for this particular application, in which not a large amount of data has

to be transferred and the user can interact closely with the device. Additionally, NFC enables simple and fast setup between devices, achievable by "tapping" on the device for establishing the connection.

Near Field Communication (NFC)

Figure 4.5: Logo for NFC-enabled devices

NFC is a communication protocol that enables point-to-point wireless connection of two devices when placed at a small distance between each other (generally smaller than 10 cm). It makes use of electromagnetic induction between two loop antennas (one in each device) and operates at 13.56 MHz (part of *industrial, scientific and medical (ISM)* radio bands).
In principle, communication is achieved by sharing a magnetic field between the coils of the two devices (as shown in figure 4.6). One of the devices, called the *reader*, generates the magnetic

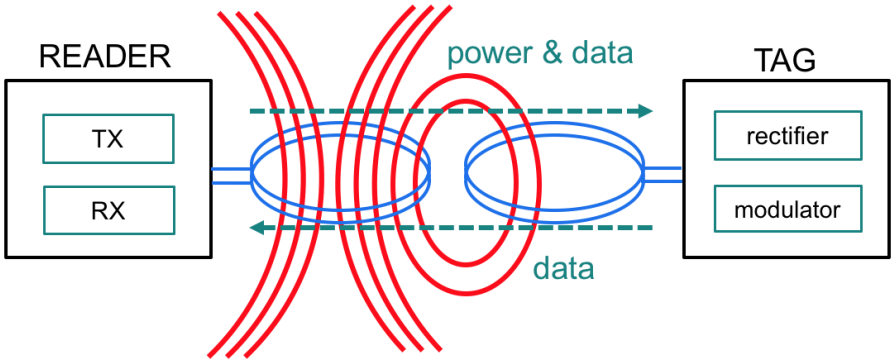


Figure 4.6: NFC communication working principle

field, which through mutual coupling induces an electric current in the second device, called *tag*. The tag, through rectification of the RF signal, generates a DC voltage for powering itself. Then, modulating the same signal (e.g. using ASK or PSK modulation), information can be exchanged between reader and tag. Furthermore, information is encoded in a standard format called NFC Data Exchange Format (NDEF) to make data easily readable by both devices.

NFC Module: AS3955

NFC functionality can be implemented on a PCB device using two components: (1) an *NFC tag* and (2) a *loop antenna*.

In this work, the NFC tag has been chosen in order to possess three key features: (i) standard interfacing protocol (SPI or I^2C), (ii) energy harvesting capabilities, and (iii) compatibility with loop antennas on PCB. Between all options on the market the AS3955, by AMS®, has been chosen for the implementation. By data-sheet specifications (summarised in table 4.3) it fulfils all three quoted requirements. Also, the AS3955 had shown in previous work [52] good results in fulfilling the harvester function. Indeed, thanks to available testing results, the possible amounts of harvested power achievable with this device were clear. This allowed, in conceiving the design, a more realistic power-budget calculation.

A block diagram of the device is shown in 4.7. The AS3955 is an NFC tag that can be used in

AS3955 Specifications	
Operating frequency	13.56 MHz
Bit rate	106 kbps
Memory	4kbit EEPROM
Energy harvesting	up to 5mA @ 4.5V
Interface	SPI and I^2C slave
Supply range	1.65V - 5.5V
Internal $C_t$une	Yes (45 pF)
Package dimensions	3mm x 3mm x 0.9mm

Table 4.3: AS3955 Specifications

both active and passive mode. In active mode the tag transmits using external power supply and can act as reader (start a transmission). In passive mode it is powered at the presence of a magnetic field generated by an NFC reader (e.g. a smartphone). Also, when in the RF field, the AS3955 can provide a regulated voltage to external circuitry in the range 1.8V-4.5V. The current provided by the regulation, therefore the power generated, is determined by: the field strength, the dimensions of the antenna and its Q factor.

For exchanging data with the reader, the AS3955 uses a NFC Forum Type 2 Logic that encodes data packets stored in NDEF format for transmission. In this application, for sending measurement data to a smartphone, values are written in the device EEPROM through the

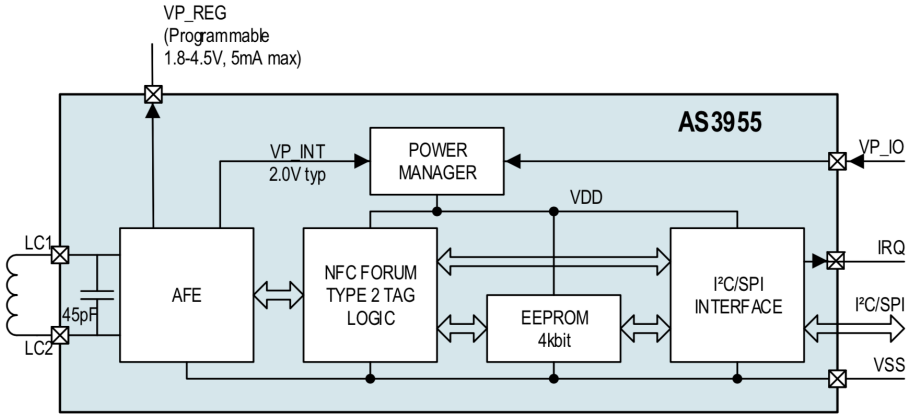


Figure 4.7: Block diagram of the AS3955

I^2C/SPI interface. Then, activating the communication, all EEPROM data can be transferred through encoding and modulation to the smartphone.

Antenna Design

An appropriate coil needs to be connected to the terminals of the AS3955 for enabling communication. Its inductance needs to be calculated before designing in order to resonate at 13.56 MHz in parallel with the internal capacitance of the AS3955 (C = 45pF). This is done in order to minimize at the frequency of interest the impedance of the system, hence maximising current and voltage delivered to the tag.[1] Knowing the expression of the resonant frequency ($\omega_0 = \frac{1}{\sqrt{LC}}$) the value of inductance can be calculated as follows:

$$L = \frac{1}{(2\pi f_0)^2 C} = \frac{1}{(2\pi(13.56 * 10^6))^2 * (45 * 10^{-12})} = 3.061\mu H \tag{4.1}$$

Given the value of inductance, a coil can be designed for obtaining that exact value, using different shapes and materials depending on the application requirements. In this work, it has been decided to design the antenna directly on the PCB as rectangular loop coil. The inductance value of such an antenna can be calculated using equation 4.2, with μ_0 magnetic permeability of free space, N number of turns, d_{out} outer diameter of the coil, and d_{in} inner

[1] For further explanation please refer to [53]

diameter of the coil.

$$L_{ant} = 2.34 * \mu_0 * N^2 * \frac{\frac{d_{out}+d_{in}}{2}}{1 + 2.75(\frac{d_{out}-din}{d_{out}+d_{in}})} \tag{4.2}$$

A more practical implementation of this equation can be found in antenna design tools. For this design, an online tool called NFC Design Navigator [54] has been used (shown in figure 4.8). This tool lets the user choose parameters of length, width, gap between tracks, track thickness and number of turns, and calculates the resulting inductance. Additionally, the obtained design can be exported in Gerber format for later use in PCB CAD tools. In tuning antenna's parameters care has been taken in order not to increase its intrinsic resistance, thus reducing its Q factor.[2].

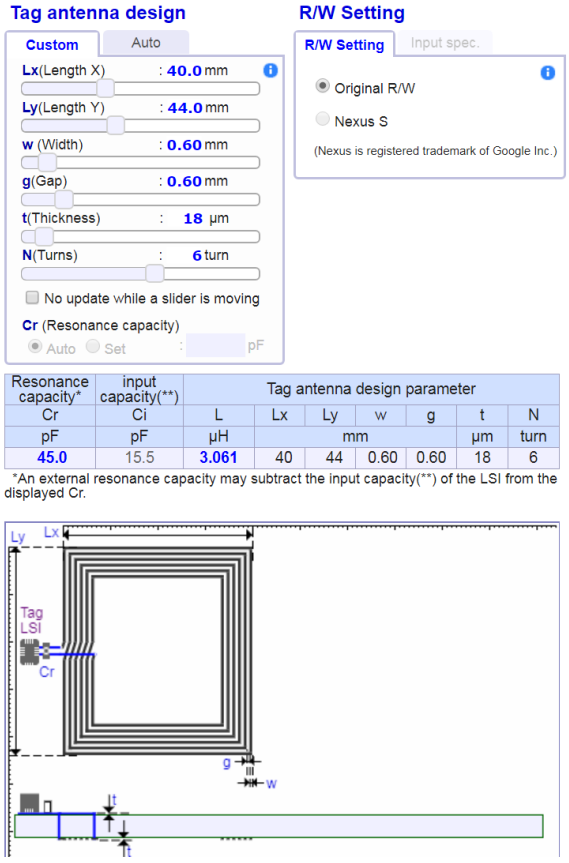


Figure 4.8: NFC Design Navigator, loop antenna online design tool

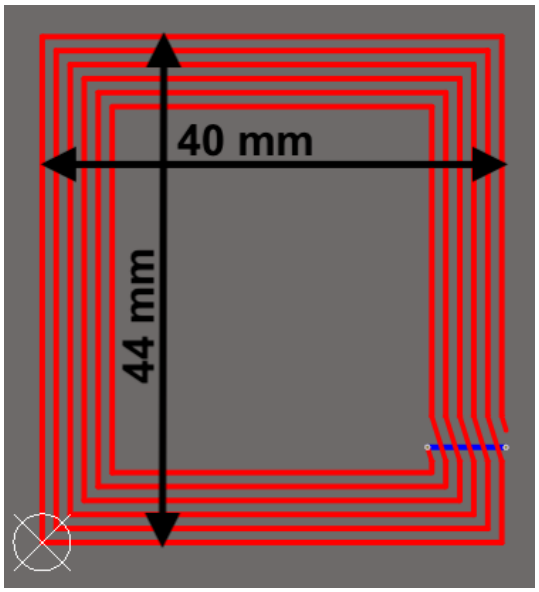


Figure 4.9: Antenna design on Altium Designer®

An antenna design is usually verified by using a High Frequency Structural Simulator (HFSS). In this way parameters like the Q factor can be estimated, allowing a more conscious evaluation of

[2]The Q factor of a non ideal inductor is defined as $Q = \frac{\omega L}{R}$, with ω radian operating frequency, L inductance and R resistance of the coil

system's capabilities. Although, learning how to use RF circuits CAD tools has been considered outside the scopes of the project, therefore it has been decided not to simulate the antenna. The parameters choice for the final antenna design is shown in 4.8.

4.1.3 Supply

As already presented, remote powering through the NFC chip has been chosen as one source of energy for the system. Although, this method only supplies the device when illuminated by the reader's electromagnetic field. To provide a steadier source of charge it has been decided to combine NFC with another energy scavenger, in a so-called *hybrid harvesting* system.
As described in section 2.2, for implementing an energy harvesting system we need: (i) an energy generator, (ii) conditioning circuitry and (iii) a storage device.

Harvester

For exploiting the generation of charges of sources like *piezoelectric* transducers, *thermoelectric* or *solar* cells, circuits called *energy harvesters* are employed. These are different types of DC-DC converters, that, depending on the type of input source, give as output a regulated voltage for supply (generally between 1.6V and 5V). Indeed, by definition, *DC-DC converters* take as input a source of current at a certain voltage level and convert it to another. If the output is lower than the input they are referred to as *step-down*, or *buck*, converters, and in the opposite case (output higher than input) they are called *step-up*, or *boost*, converters. A mix between the two types is also possible, so-called *buck-boost* converters, in which the input supply can vary widely, and be both higher and lower than the regulated output.
All these types of power managing circuits are present on the market as off-the-shelf components, with various features and for different applications. Analog Devices® provides a wide selection of energy harvesters. Further, thanks to an online selection table [55] it is easy to find the right component for one's application.
In this work, rather than selecting one type of transducer and focusing on optimising the gathering of energy, it has been decided to realise a multiuse scavenging interface. This has been done

in order to assess with the same device various possible combinations of harvesting depending on its target use. Indeed, the choice of a transducer is very application dependent, since it is related to what form of energy the final device will withstand the most (e.g. mechanical, magnetic, thermal, etc.).

For obtaining this feature, the LTC3588-1 has been chosen as best option. Its very large input operating range (2.7V - 20V) enables the use with various sources, or also with multiple sources combined. The LTC3588-1 (block diagram in figure 4.10) is a buck converter that also provides a full-bridge rectifier input, possibly allowing the interface with piezoelectric, solar, magnetic and thermoelectric transducers. Its output is a selectable regulated voltage with an output current up to 100mA. Also, its output logic notifies with a "PGOOD" signal when the voltage has achieved regulation.

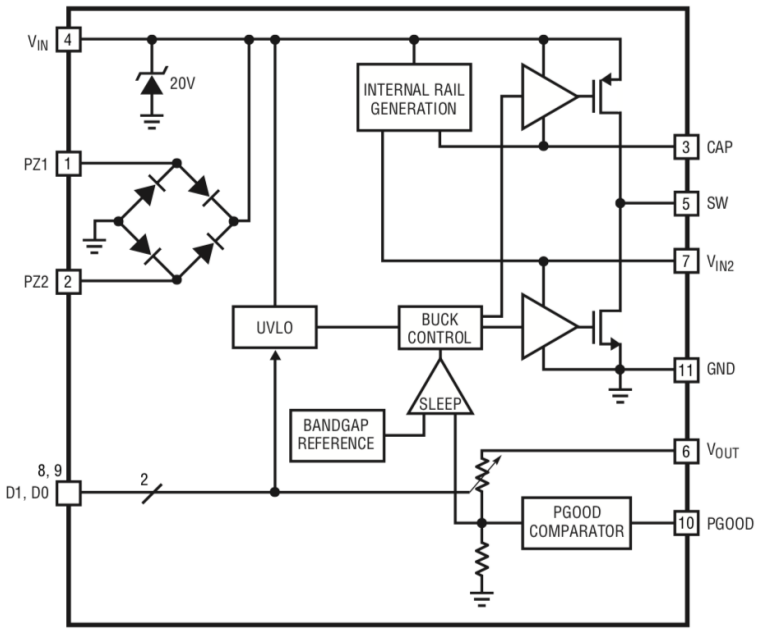


Figure 4.10: Block diagram of the LTC3588-1

Storage

To store the energy obtained by the harvester iy can be used either a *battery* or a *supercapacitor*. The advantages of using one with respect to the other have been explained in chapter 2.

For this implementation a small supercapacitor has been chosen as energy storage. Two parameters are essential for selecting the right component for the scope: capacitance and voltage. The

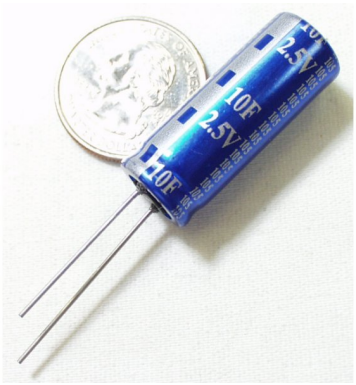


Figure 4.11: Example of supercapacitor

capacitance tell us how much charge the device can store, and in commercial supercapacitors it can vary between hundreds of millifarads up to hundreds of farads. The voltage is the level in volts that the capacitor holds when full of charge, and is usually between 2.5V and 3V for portable devices.

To evaluate the amount of charge needed for the application, considerations about the power consumption of the devices on board need to be done. Moreover, the trade-off between supercapacitor charging time and usage time of the device needs to be taken into account. Therefore, the choice of the size and features of the device has been done after having decided the final structure of the system.

Conditioning

In addition to its use for gathering energy from harvesting sources, conditioning circuitry has to be designed to utilise harvested charge properly, at the right time, and without waste.

Since it has been decided to store energy in a supercapacitor, a step-up converter has to be placed between the storage and the main supply line. Indeed, the voltage on the capacitor can be lower than the main supply (e.g. 3.3v) and needs to be converted to a higher regulated value. Also, it needs to be fully discharged in order to exploit all harvested energy. Therefore, we need a regulator able to provide a stable output also when the input voltage is at very low levels. Finally, care has to be taken in choosing a conversion with the highest possible efficiency, not to waste scavenged energy.

For this purpose it has been chosen the LTC3105 (block diagram in figure 4.12), a high efficiency

step-up DC-DC converter with low input start-up voltage (250mV). As shown in figure 4.13,

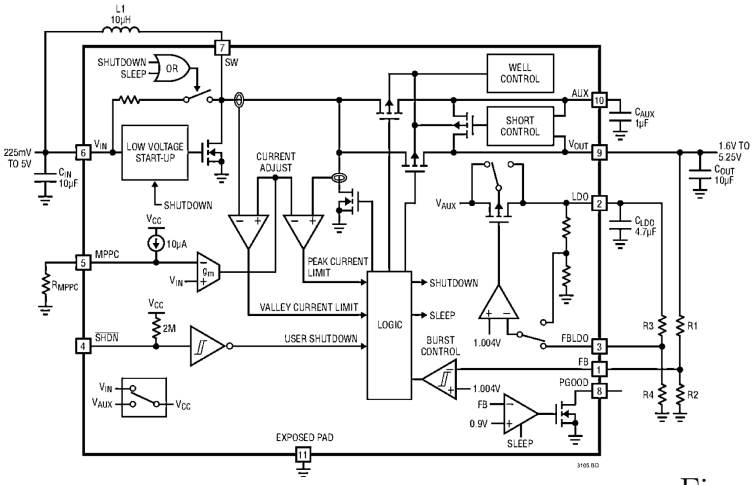


Figure 4.12: Block diagram of the LTC3105

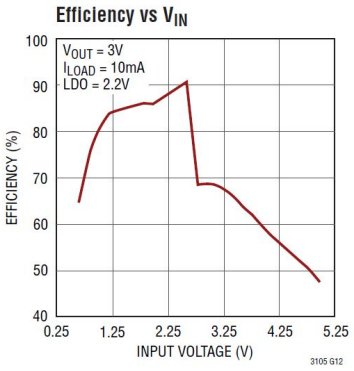


Figure 4.13: Efficiency vs V_{IN} line graph

the device can provide a 3V output with an efficiency above 80% with inputs between 1 and 2V. Moreover, it has an auxiliary Low-dropout regulator (LDO) that generates another output, regulated at a lower voltage and with a current limit at 6mA. Both output levels are selectable using external resistors.

4.1.4 Concept designs

With the chosen devices two concept designs have been envisioned (shown in figure 4.14 and 4.15). As already mentioned, the general aim has been to realise a battery-less device for performing electrochemical measurements and impedance analysis. Although, for making sure the final device could function also for other, not foreseen, applications the designs included optional blocks and selectable components.

The main difference between the two concept designs is in the impedance analysis block and, consequently, in the expected performance of impedance analysis measurements. Both structures are completely original in the combination of devices, inspiration has been taken by suggested applications of the different components.

In both designs AS3955 and LTC3588-1 have been combined for realising a hybrid energy harvesting block. These two devices gather energy from the respective sources and store it in the supercapacitor. Schottky diodes have been placed in between the harvesting modules and

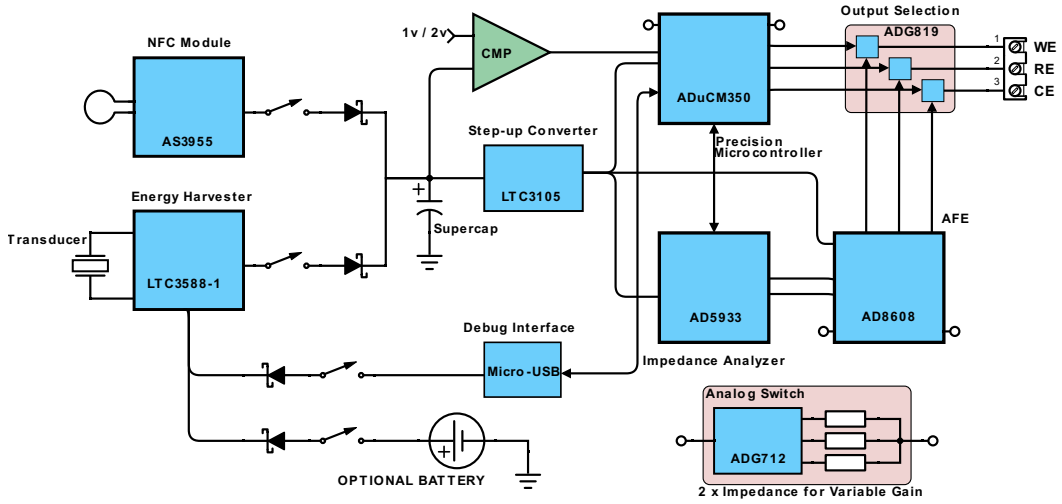


Figure 4.14: First PCB concept design

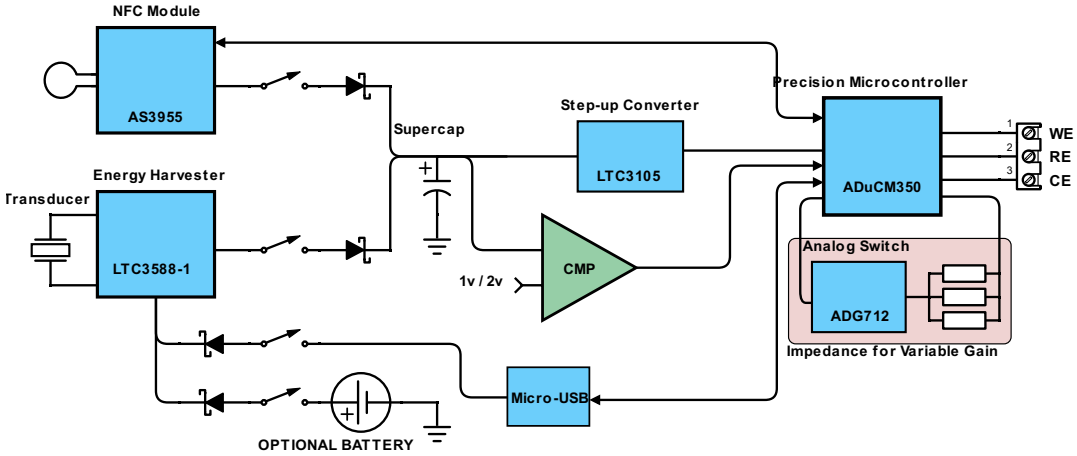


Figure 4.15: Second PCB concept design

the supercapacitor, in order to avoid inverse currents. The energy harvester also includes as possible sources: a USB supply and an optional coin cell battery. These two have been added to eventually power the device for debugging and testing. The outputs of the NFC module and the harvester have been set for providing 2.5V in order to charge the supercapacitor up to around 2.2V (2.5V minus the forward voltage of the diodes).

Then, the step-up converter has been placed in both designs between supercapacitor and main supply. This ha been used for generating the 3.3V output and power the system. Also, a comparator has been used to check the voltage level of the supercapacitor and send a signal to the μC at fixed thresholds. The functioning of this block will be further explained in what follows.

Finally, in the first design the ADuCM350 has been combined with the AD5933 and an additional AFE. Further, switches and multiplexers have been used to tune variable gains and select the interface with the sensor.
Then, in the second design, the ADuCM350 has been utilised by itself, with analog switches for selecting at run-time the TIA gain.
Of the two options it has been decided to proceed in the design of the second device. Even if this choice meant not meeting completely the initial specifications (in terms of frequency range), the second device could benefit of a much lower power consumption with respect to the first. Indeed, with the extra impedance analyser and AFE it would have been difficult to realistically power the whole device in a reasonable time using NFC and energy harvesting.

Power budget

For sizing the storage capacitor and evaluating the possible performance of the device, the power consumption of the device has been evaluated. In this calculation both NFC module and energy harvester have been discarded, since they are powered autonomously in this implementation.
The parameters that have been taken into account are: (i) charging time, (ii) current consumption of the system, (iii) usage time and (iv) efficiency of the step-up conversion.
Charging time has been evaluated as the time needed for the supercapacitor to reach full charge with the available harvesting sources (5mA maximum). A 2.5V supercapacitor has been chosen in order to match the output of the harvesting modules. Subsequently, a table has been filled in with all charging times (up to 2.2V) for different options of supercapacitor size (in figure 4.4). Times have been calculated assuming constant charge ($t = \frac{C*V}{I}$, with t charging time, C capacitance, V voltage, and I charging current).

Size	Charging Time
0,2 F	88 seconds
0,5 F	220 seconds
1 F	440 seconds
2 F	880 seconds

Table 4.4: Supercapacitor size vs charging time

In terms of consumption the most relevant contribution is given by the ADuCM350, which consumes a maximum of 8mA at measurement run-time. With respect to 8mA, other components consume a negligible amount of current and have been excluded from the computation.
Regarding the working time of the device, it has been evaluated in the range between 30 seconds and 1 minute depending on the type of measurement. For the sake of the evaluation 30 seconds has been used as reference.
Then, the efficiency of the step-up conversion has been approximated to 85% for inputs between 1.25V and 2.25V and 67% for between 0.25V and 1.25V (values extrapolated from figure 4.13).
Finally, the charge needed in the supercapacitor for performing one measure has been calculated. Energy conservation (equation 4.3) has been taken into account for determining the relationship between needed charge and stored charge (i.e. for considering the efficiency of the step-up). Q_{out} is the charge needed for the supply current ($8mA * 30s = 0.24C$), Q_{in} the charge on the supercapacitor, V_{out} the output voltage at 3.3V, V_{in} the input voltage , η the efficiency of the step-up and t time.

$$V_{out} * Q_{out} = \eta V_{in} * Q_{in} \tag{4.3}$$

Two values of Q_{in} have been calculated for the two ranges of efficiency at mid-range voltage (67% @ 0.75V and 85% @ 1.75V). For the higher efficiency range $Q_{in} = 2.21 * Q_{out} = 0.53$C and for lower efficiency $Q_{in} = 6.57 * Q_{out} = 1.58$C.
To conclude, this charge has been correlated to the voltage variation on the supercapacitor, for different supercapacitor sizes ($Volt = \frac{Coulomb}{Farad}$). This has been done in order to choose the right device for operating at higher efficiency range. A summary of all voltage variations for different capacitor sizes is shown in table 4.5, with full charging times as reference.

$\eta = 85\%, Q_{in} = 0.53C$			
C	0.5F	1F	2F
ΔV	1.06V	0.53V	0.26V
t	220s	440s	880s

$\eta = 67\%, Q_{in} = 1.58C$			
C	0.5F	1F	2F
ΔV	3.16V	1.58V	0.79V
t	220s	440s	880s

Table 4.5: Capacitor size vs Voltage

A 1 Farad supercapacitor has been chosen as viable option for operating at high efficiency range, even if limited in terms of fast charging time.

Final design functionality

Therefore, the final device has been designed to operate as follows:

1. The energy harvesting modules start charging the supercapacitor;
2. The step-up converter turns on the system as soon as its start up voltage level is reached at the input;
3. The μC, as soon as it is turned on, goes to hibernate mode (all peripherals off waiting for a wake up signal);
4. The comparator checks the voltage level on the supercap and notifies the μC as soon as a threshold is reached;
5. The μC starts a measurement and write results on the NFC module;
6. The user reads measurement results using an NFC reader.

4.2 Development and Assembly

4.2.1 CAD Project

The project of the device has been developed using Altium Designer®.
First, all schematic and PCB libraries have been created and filled in with *schematic symbols* and *footprints*. Various footprints have been created according to devices' guidelines and some have been modified in order to facilitate the ensuing soldering and assembly stages. Then, a schematic project has been created and, following all manufacturers directions, conditioning components have been added to the project and sized according to their scope. Finally, all components have been linked between each other and to the power lines.
Finished the schematic, components have to be placed in the final layout of the board. The software allows to generate automatically a board layout from the schematic. This simply

adds to a layout document all footprints linked to the schematic symbols, without placing or connecting them.

The board has been chosen to be a double layer PCB with a dimension of 10 cm x 5 cm. Devices have been placed on the board making sure sources of noise (e.g. oscillators, switching inductors) were away from the AFE and from the connections to the sensor. *Tracks* and *vias* of different sizes have been used to connect the components. The width of each track has been evaluated according to the current the path had to drive.

Extra care has been taken in the placement and the connections of the ADuCM350. Indeed, the only available package of the device is a 120-pins *ball grid array (BGA)*, with a pitch of 0.2 mm between pads. This required to pay special attention in using only two layers for making all connections and in placing vias under the device. The result of the placing are shown in figures 4.16 and 3.3. Details of both schematic and layout can be found in the appendix of this document.

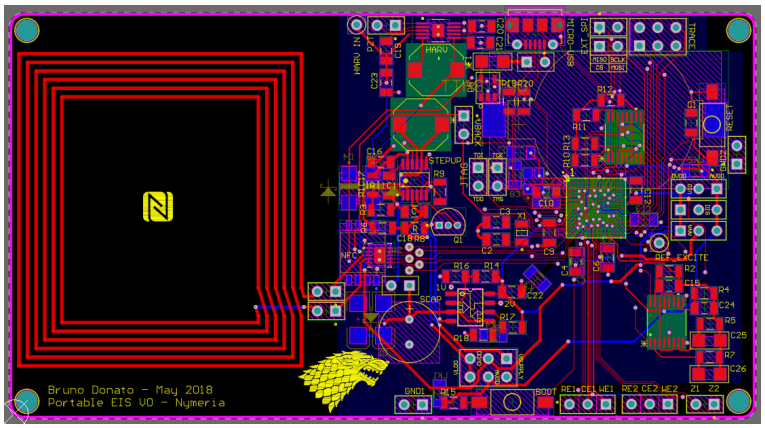


Figure 4.16: Final board layout

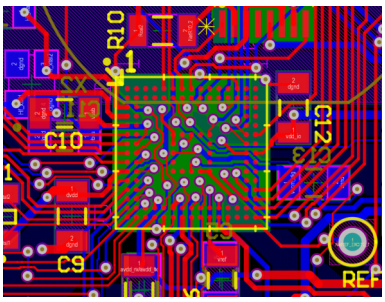


Figure 4.17: BGA detail

A *layout versus schematic (LVS)* check has been done on the final layout to see if all made connections match the circuit diagram. Also, a *design rule check (DRC)* has been run to see if all production directives had been respected. Design rules have been manually compiled in the software in order to match the ones of the chosen manufacturer (Eurocircuits® Gmbh). Finally, manufacturing outputs have been exported in the form of *Gerber* and *NC Drill* files and uploaded online on Eurocircuits® PCB Calculator [56] for proceeding with board production. With the PCB also a *stencil* has been ordered, selecting only the top layer of the board. This is used to help spreading *solder paste* in the assembly stage. In addition, a PCB frame (*eC-*

registration panel) has been added for using it as guidance in the alignment with the stencil.

4.2.2 Assembly

The bare manufactured board received from Eurocircuits® looks like in figure 4.18 and 4.19.

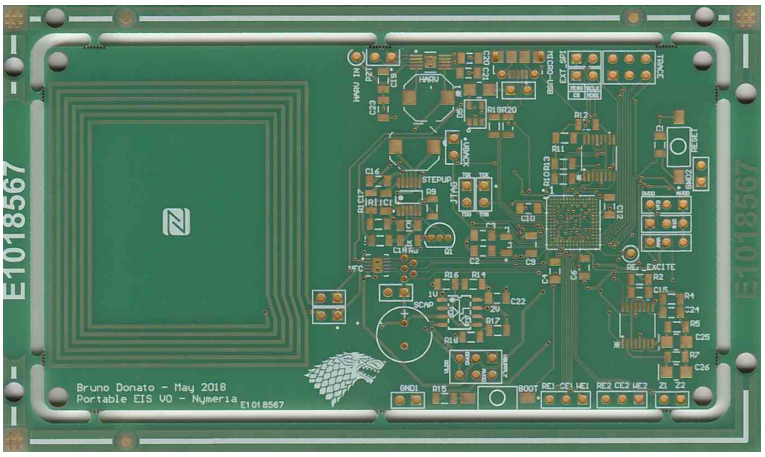


Figure 4.18: Bare board front side

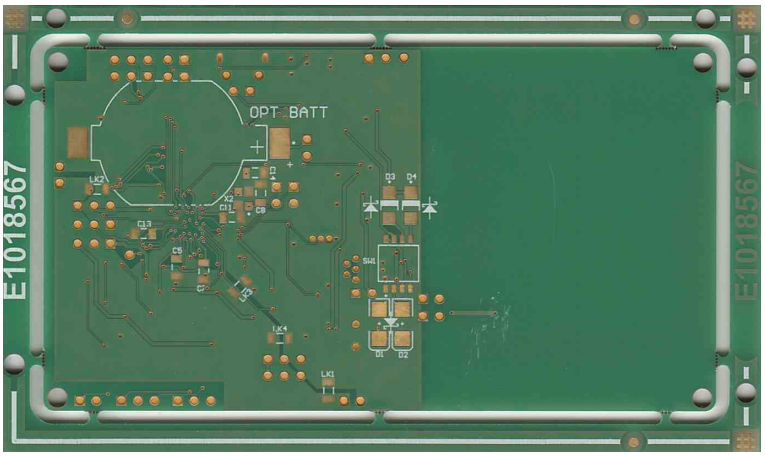


Figure 4.19: Bare board back side

On it, all components have been placed and soldered. For doing this two different methods have been used. On the back side, components have been soldered with a traditional soldering iron. On the front side, that hosts all surface-mounted packages and the BGA component, soldering has been done using *reflow soldering*. This second method consists in three main steps: (i) solder paste deposition, (ii) components placing and (iii) heating in a *reflow oven*.

First, solder paste, a sticky mixture of a resin called "flux" and small metallic particles, has

been spread on the board for covering all exposed pads. This has been achieved using the manufactured stencil (figure 4.20). Then, components have been placed with the aid of a pick and place machine (figure 4.21). Finally, the board has been placed in the reflow oven and subjected to a heat. The thermal profile has been defined according to the specifications of the chosen solder paste[3] (Chip Quik® SMDLTLFP10T5) and oven parameters have been set accordingly.

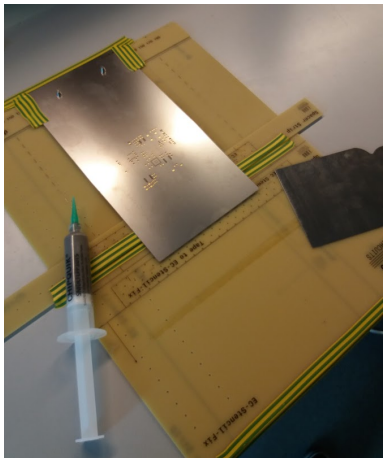


Figure 4.20: Solder paste deposition

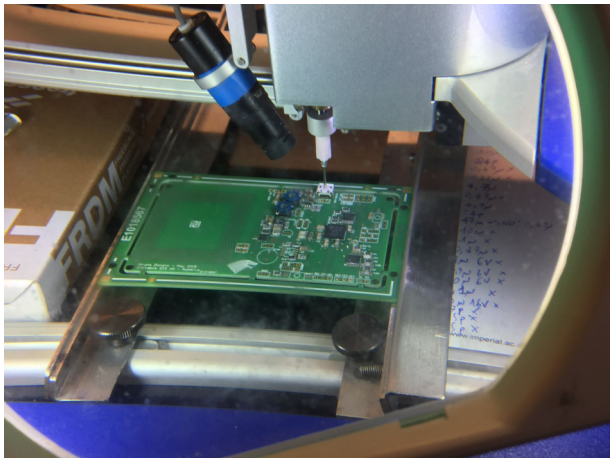


Figure 4.21: Components placing

Finalised the reflow step, all through-hole components and everything on the back side has been manually soldered. The final result of the assembled PCB is shown in figure 4.22.

Figure 4.22: Assembled PCB

[3]Selecting the right solder paste is crucial for achieving a good final result, especially when covering very small pads like the ones of the BGA component. Interested readers please refer to [57]

4.3 Final Testing

Before any specific test, the board has been checked with a microscope, and all relevant pads and pins have been tested using a multimeter for finding eventual short circuit paths caused by the reflow soldering. When needed, shorts have been removed using a syringe needle while gently heating the section with a heat gun.
Then, in sequence, the functionality of all main blocks has been accessed. First, supply and wireless communication, and then the μC and AFE.

4.3.1 Energy Harvesting Block

The LTC3588-1 has been tested by checking the voltage at the output while applying a stimulus at the input using piezoelectric diaphragms. The chosen transducers produce a maximum voltage of 15V. A parallel of three diaphragms has been used to keep the generated voltage in the operating range and increase the current output of the transduction. Figure 4.23 shows how, by repeatedly pressing on the diaphragm, a voltage of 2.5V has been generated at the output.

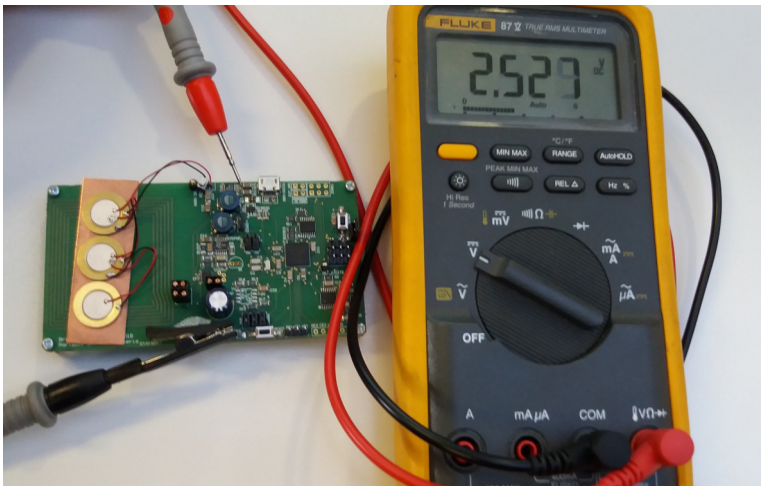


Figure 4.23: Piezoelectric harvesting test

4.3.2 Wireless Communication Block

The NFC communication has been tested first between NFC chip and reader (i.e. an android smartphone). Placing the reader on top of the board, at a distance of around 3 cm, the device information could be read through an android app (*NFC Reader*, by Adam Nybäck), as shown in figure 4.5. The test confirmed that the loop antenna and the device connections had been

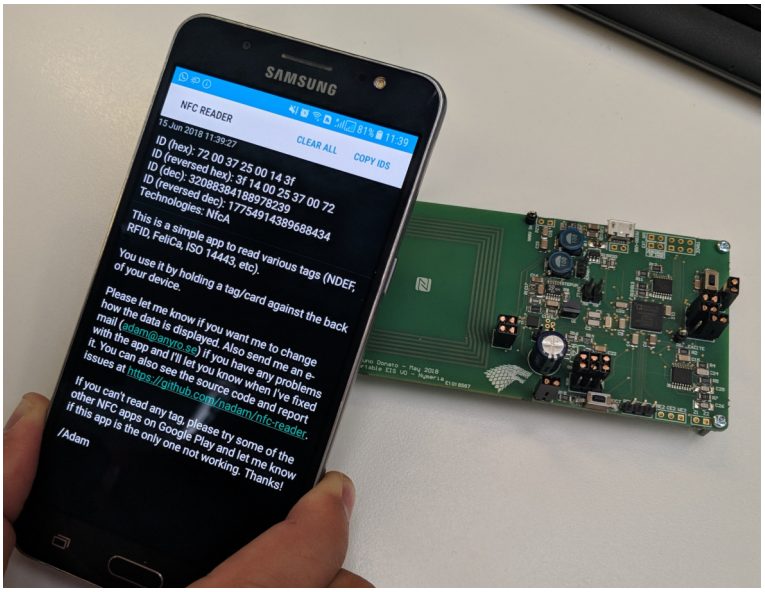


Figure 4.24: NFC tag test

properly designed. Then, the use of the internal EEPROM of the AS3955 has been tested. For this the SPI communication has been configured on the μC as described on the datasheet of the NFC module and debugged using an external logic analyser (result in figure 4.25). Then

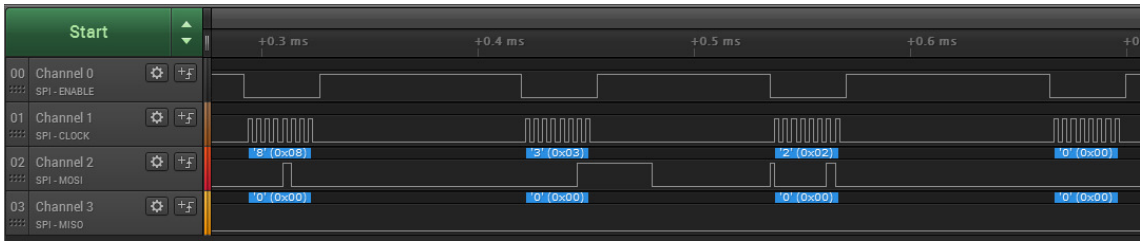


Figure 4.25: SPI to NFC test

sample blocks in memory have been written through SPI and read from the smartphone (this time using *NFC TagInfo* app, by NXP®).

SPI has also been used to configure successfully the AS3955 mode for energy harvesting and select the output voltage of the on-chip regulator to 2.5V.

4.3.3 Microcontroller

All peripherals of the ADuCM350 needed for the application have been tested. For this, simple firmware has been developed for successively testing: GPIO, UART, SPI. The code has been downloaded on the μC flash memory by using serial communication. Indeed, unfortunately, it has not been possible to communicate to the μC through the JTAG debbuging interface, probably due to a driver issue of the device used for programming (a JLink by Segger®). All firmware has been developed using IAR Systems® Embedded Workbench, the IDE supported by the manufacturer. Code has been compiled using a proprietary toolchain contained in the industrial development kit (IDK) provided by Analog Devices®. GPIO has been tested by toggling external PCB pins while probing with a logic analyser. UART functionality has been validated connecting to the PCB a UART-to-USB cable and receiving results of "printf" statements on a serial terminal on the PC. SPI has been debugged first through four pins on the PCB and then by probing the tracks between the μC and the AS3955.

4.3.4 Measurements

Finally, to validate the functionality of the whole device, amperometric measurements and impedance analysis have been performed. Two code routines have been developed (in appendix) for chronoamperometry and impedance spectroscopy. Measurements have been performed by connecting the sensor output of the PCB to a breadboard as shown in figure 4.26. Results have been gathered through UART communication from a PC and plot with Matlab® for demonstration purposes.

Before any measure, the gain of the TIA has been tuned for avoiding saturation of the AFE and obtaining a measurable signal. This has been done by selecting the right TIA feedback impedance block using the analog switches on board. This evaluation has been made considering the expected input current of the AFE.

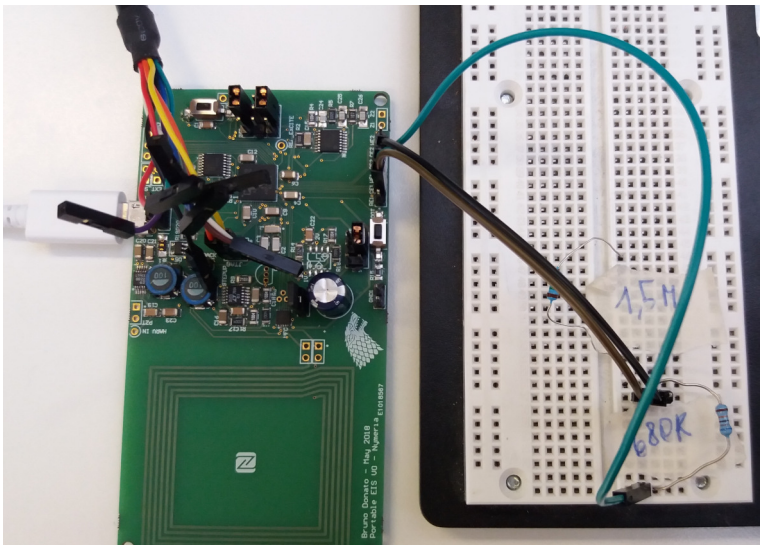


Figure 4.26: PCB measurement testing setup

Chronoamperometry

Chronoamperometry has been mimicked by connecting a 680kΩ resistor in parallel with a 1.5MΩ potentiometer. By varying the impedance value while applying a fixed voltage (600 mV) an increasing current in time has been obtained as expected. The results of this test are shown in figure 4.27.

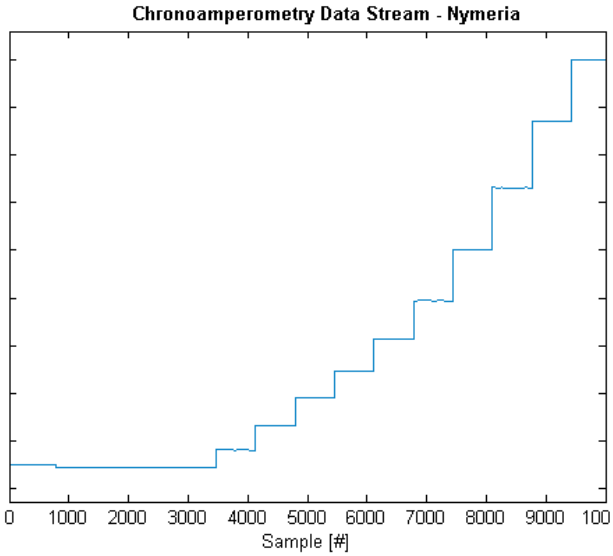


Figure 4.27: Chronoamperometry of 680kΩ resitor and varying potentiometer

Impedance Spectroscopy

Finally, impedance spectroscopy has been run on a 680kΩ±1% resistor. The amplitude of the stimulus has been set to 10mV and frequency has been swept between 100Hz and 10kHz (5 points per decade). In figure 4.28 is shown the series of sinusoidal pulses generated by the PCB during impedance spectroscopy.

Figure 4.28: Impedance spectroscopy stimulus

The results gathered from the UART console and plot in the form of amplitude and phase Bode diagram are shown in figure 4.29.

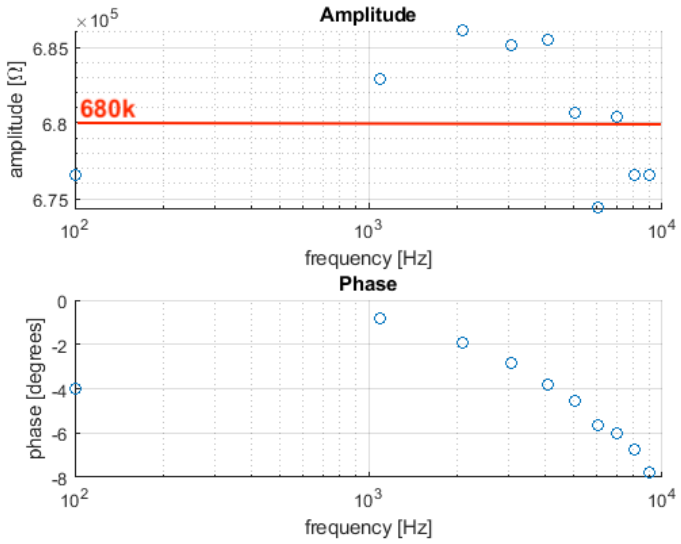


Figure 4.29: Bode diagram of 680kΩ resistor measurement

As shown, the obtained value of amplitude is within the expected range. The phase shows a capacitive behaviour increasing with frequency, probably due to the parasitics of the connection with the breadboard.

Furthermore, consecutive impedance spectroscopy measurements have been performed with the device disconnected from any power source. Using a fully charged supercapacitor three complete impedance spectroscopy measurements have been performed (100Hz-10kHz, 10 points/decade).

4.3.5 Measure transfer

Last, a two frequencies impedance spectroscopy measurement has been performed on the same resistor and measurement results have been written on the NFC tag and read from the smartphone. In this way the functionality of the device has been validated from measurement to user feedback. The measured values of magnitude and phase are shown in figure 4.30 with their value converted in hexadecimal string. A screenshot of the smartphone app showing the EEPROM memory content containing the same results is shown in figure 4.31.

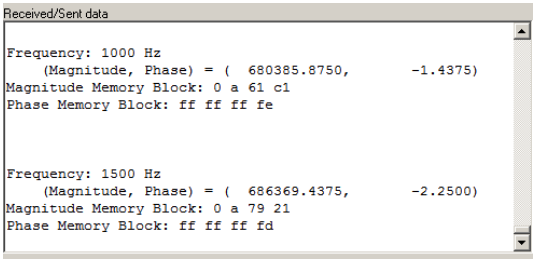


Figure 4.30: Two frequencies measurement results

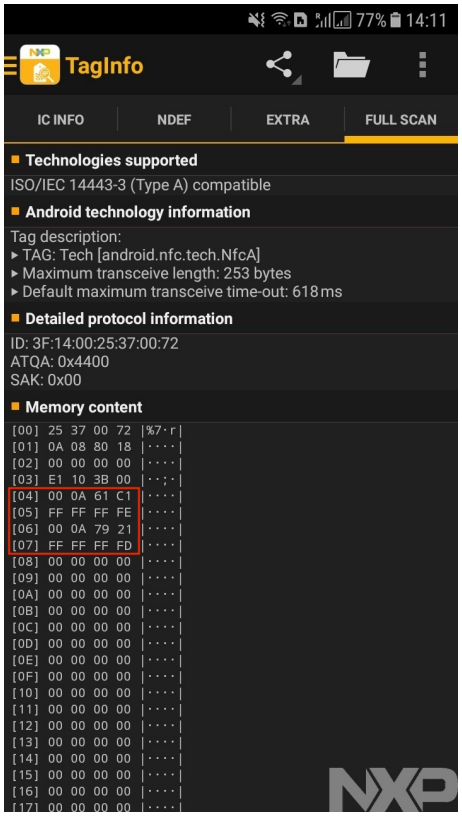


Figure 4.31: Smartphone app showing the EEPROM content

4.3.6 Electrochemical measurements

Electrochemical measurements have been attempted connecting the PCB to available commercial sensors in PBS solution. Unfortunately, due to an excessive equivalent capacitance of the sensors, destabilising the AFE, any measurement could be performed. Measurements using

other sensors of smaller capacitance are postponed to future work.

To evaluate the maximum load capacitance of the AFE various R-C parallels have been tested varying successively the value of the capacitor. Stable measurements have been achieved with capacitances smaller than 10nF.

4.4 Chapter Summary

A portable battery-less PCB for electrochemical measurements and impedance analysis has been conceived, designed, assembled and tested. It's functionality has been validated with tests verifying each PCB block: energy harvesting, wireless comunication, microcontroller and AFE. Chronoamperometry and impedance spectroscopy have been performed on a reference through-hole resistor. Measurement results have been successfully read from a smartphone through NFC.

Chapter 5

Conclusion and Future Work

In this thesis I assessed the possibility of realising a device for portable bio-sensing and impedance analysis, that may allow the detection of different types on bio-markers by using various sensors. The work has been carried out during a nine months exchange project at the *"Centre for Bio-Inspired Technology", Imperial College London (UK).*

As planned, a prototype for bio-sensing and impedance analysis has been assembled. Also, obstacles and limitations that arise when implementing this type of device in portable applications have been evaluated. Furthermore, a battery-less PCB for electrochemical measurements has been conceived, designed, realised and tested. It's functionality has been verified and trial measurements have been conducted with successful results.

5.1 Discussion

In the prototyping stage, an entire measurement platform has been realised. This has been achieved making use of: (i) a microcontroller evaluation board, (ii) a pre-developed custom Analog Front-End (AFE) chip placed on a testing PCB, (iii) a PC for user interface, and (iv) a commercial sensor. The work in this first section has been focused on the development of firmware for the μC unit, the debugging of all problems related to the connection between the two boards, the development of a user interface and post-processing software on PC, and the use

of the setup for electrochemical measurements with a commercial sensor. Multiple limitations have been encountered in the prototype implementation. Concerning the μC board, main obstacles have been: restricted samples storage, the speed of stimulus-sampling synchronisation, the update frequency of the DAC output register, and noise limiting the minimum stimulus amplitude. Moreover, in the use of the AFE, platform's performances have been limited by: current range, gain accuracy, and ADC sampling frequency limitations. Nevertheless, all three types of measurement have been performed. The setup has shown satisfactory behaviour in Chronoamperometry (CA), obtaining results similar to laboratory instrumentation. On the other hand, system constraints have not allowed successful measurements in Cyclic Voltammetry (CV) and Electrochemical Impedance Spectroscopy (EIS). In the first case, this was probably due to the way the voltage stimulus has been applied to the cell, which could not trigger the reaction properly. Moreover, in EIS, the discrepancy between calculated and true gain of the system generated a variability in the measurement of amplitude. This issue could have been probably solved establishing and utilising a pre-measurement calibration routine.
Concerning the PCB, a device showing an original use of energy harvesting techniques applied to electrochemical sensing has been conceived. Two alternative conceptual block diagrams have been envisioned, and one has been chosen for development. All steps of PCB design flow have been successfully performed: components choice, schematic design, layout design, generation of fabrication outputs and compliance with design manufacturing guidelines. The bare PCB received from fabrication has been successfully assembled adopting solder paste deposition (utilising a specifically fabricated stencil), components placing with SMT placement equipment, and soldering using a PCB reflow oven. Firmware for performing CA and EIS measurements with the PCB has been developed and downloaded on the μC on-board. Board functionality has been tested resulting in all three main blocks working properly: hybrid harvesting supply, supercapacitor and step-up, microcontroller and AFE. Unfortunately, due to manufacturing and delivery delays, the conformity to the power budget and the sleep-wakeup functionality of the μC could not be assessed and had to be postponed to future work. Measurements with electrochemical medium have been attempted, but could not be completed due to difficulties in finding a compatible sensor with the AFE in use. Indeed, the integrated front-end is only

able to drive electrodes with capacitance equivalent smaller than approximately 10nF, a value much lower than the one of the available screen printed electrodes. The use of the PCB with compatible sensors has been postponed to future work. However, CA and EIS functionality has been assessed using a resistor discrete component and a potentiometer, performing both measurement successfully. Additionally, system's ability of sending measurement feedback from the device to a user's smartphone has been verified, demonstrating completely the expected device operation.

5.2 Future Work

With respect to current literature and commercial devices, the current work proposes a novel approach to the design of portable electrochemical sensing devices. It combines the ability of performing multiple types of electrochemical measurements with the advantages of having a portable battery-less device. Possible applications of such a device are both in point-of-care and in personal diagnostics. Although, further assessments have to be made for verifying all that all device's features correspond to what conceived. First of all, using the right sensor, bio-markers measurements could be performed. Furthermore, comparisons of CA, CV and EIS measurements performance could be made with laboratory instrumentation, and device's characteristics could be outlined. Then, sensing experiments with various analytes could be performed, both for amperometric and capacitive sensing. Moreover, alternative harvesting sources could be tested, such as Peltier thermoelectric generators or solar cells. Finally, focusing on a set of analytes and choosing the right type of harvesting, a new PCB device could be designed. This could be smaller and possibly on flexible substrate, to be used as wearable device for personal diagnostics in sweat, saliva or interstitial fluids sensing.

Appendix A

Code

A.1 Prototype Code: Mooncake Measurement Console

A.1.1 Firmware

```
/* ###################################################################
 **     Filename    : main.c
 **     Project     : MOONCAKE_CONSOLE
 **     Processor   : MKL26Z128VLH4
 **     Version     :  3
 **     Compiler    : GNU C Compiler
 **     Date/Time   : June 2018
 **     Abstract    : KL26Z Firmware for CA, CV and EIS
 ** ###################################################################*/
/* omitted global variable definition */
int main(void){
  /* omitted local variable definition  */
  /*** Processor Expert internal initialization ***/
  PE_low_level_init();
  for(;;){
    //***************START*******************//
    LED2_Neg();//LED feeback
    fgets(input_buffer,3,stdin);//uart communication
    LED2_Neg();//LED feeback

    switch(input_buffer[1]){
    /******************EIS*******************/
    case 'S':
      //********START EIS ROUTINE********//
      //printf("\n\n*********EIS_CONSOLE*********\n");//user feedback
      //get measurement parameters
      fgets(input_stream,5,stdin);//
      V_AMPLITUDE = atoi(input_buffer); //peak to peak amplitude
      fgets(input_stream,5,stdin);
      OFFSET = atoi(input_buffer);//voltage offset
```

```
31        fgets(input_stream,2,stdin);
32        MEASURE_CYCLES = atoi(input_buffer);//measurement cycles
33        fgets(input_stream,3,stdin);
34        init_wait = atoi(input_buffer);//initial wait time
35        fgets(input_stream,3,stdin);
36        freq_steps = atoi(input_buffer);//number of frequency steps
37
38        sine_frequency = malloc((freq_steps+1)*sizeof(uint32_t));//allocation in memory
39        for(i=0; i<freq_steps; i++){
40          fgets(input_stream,8,stdin);
41          sine_frequency[i] = atoi(input_buffer);//values of frequency for each step
42        }
43        selADC = 0; //disable internal adc for bypassing externally using ADC_in pin
44              //get chip parameters
45        fgets(input_stream,6,stdin);
46        clk_freq = atoi(input_buffer);//clock frequecny
47        fgets(input_stream,3,stdin);
48        PGA_gain = atoi(input_buffer);//pga gain
49        //set clk frequency
50        CLKADC1_SetFreqkHz((word)(clk_freq/1000));
51        //convert PGA_gain into binary
52        switch(PGA_gain){
53        case 1:
54          PGA_gain = 0b0001;
55          break;
56        case 2:
57          PGA_gain = 0b0010;
58          break;
59        case 3:
60          PGA_gain = 0b0011;
61          break;
62          /* omitted other cases */
63        }
64  //MOONCAKE CFG REGISTERS
65        /*<15>*/selREsource1 = 1;
66        //if =0 VRE1=V_DAC1; if =1, VRE1=VREext
67        /*<14>*/selAL = 1;
68        //if =1 the Aluminium electrodes are active; if =0 the ISFETs are active
69        /*<13>*/single = 1;
70        //if=1, single electrode is connected; if =0, two electrodes
71        /*<12>*/selADC = 1;
72        //if =1, system out is connected to the ADC; if =0, the system_out is connected to pin "systemout"
      through a voltage buffer;
73        //in this case the ADC_in is accessible through the ADC_in pin; Also if selADC=1, the clksystem=clkADC
      /11; if selADC=0, then clksystem=clkADC;
74        /*<11>*/selWE1 = 1;
75        //if=1, WE1 is connected; if =0, WE2 is connected; if single=0, then selWE1 doesnt care ***ERRATA
      CORRIGE 1st VERSION***
76        /*<10>*/selREsource2 = 1;
77        //if =0, VRE2=V_DAC2; if =1, VRE2=VRE1
78        /*<9:0>*/ //use for RE1 DAC initialisation
79        //**********CFG REGISTERS*********//
80        SPI_ADC_middle= (selREsource1<< 15)| (selAL <<14)| (single<<13) | (selADC<<12) | (selWE1<<11) | (
      selREsource2<<10) | 0b0000000000;
81        SPI_DAC_middle= (0b00<<14) | (PGA_gain<<10) | 0b0000000000;
82        SPI_ADC_top= (selREsource1<< 15)| (selAL <<14)| (single<<13) | (selADC<<12) | (selWE1<<11) | (
      selREsource2<<10) | 0b0000000000;
83        SPI_DAC_top=(0b00<<14) | (PGA_gain<<10) | 0b0000000000;
84        //*********SEND CFG*************//
85        Send_Cfg(MyGPIO1Ptr, SPI_ADC_middle, SPI_DAC_middle, SPI_ADC_top ,SPI_DAC_top);
```

```
86      //allocate memory for data
87      data = malloc((MEASURE_CYCLES)*sizeof(uint16*));
88      /* omitted check of data allocation */
89      //flash initialisation
90      FLASHPtr = IntFlashLdd1_Init(NULL);
91      //**************LUT GENERATION**************//
92      Gen_LUT();
93      //Set initial voltage for stabilisation
94      voltage_step = OFFSET/DAC_LSB; //convert mV to LSBs
95      DAC_PDD_SetData(DAC0_BASE_PTR,(uint16_t)(voltage_step),0U);
96      flash_index = 0;
97      for(step_index=0; step_index<freq_steps; step_index++){
98        lut_step = 1;
99        //calculate counter ticks from frequency
100       ticks = (TU2_CNT_INP_FREQ_R_0*1000)/(sine_frequency[step_index]*(LUT_LENGTH/lut_step)); //(ex. 1MHz
      *1000/(100mHz*720))
101       //*****************INITIALISATION*****************//
102       //SPI_ADC_middle 0x01
103       SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL0, 0b1);
104       SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL1, 0b0);
105       SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL2, 0b0);
106       cycle = 0; //default values
107       phase = 0;
108       sample = 1;
109       TU2Ptr = TU2_Init(NULL); //TU2 initialisation
110       TU2_SetPeriodTicks(TU2Ptr,(uint32_t)ticks); //set timer ticks
111       end_measure_flag = FALSE;//frequency cycle flag down
112       start_DAC_flag = TRUE; //start DAC
113       while(end_measure_flag == FALSE){
114         while(EOC_flag != TRUE){
115         }
116         EOC_flag = FALSE;
117         //data from ADC
118         SM1_TransceiveBlock(SM1_DeviceData, &SPI_ADC_middle, &sample, 1);
119         data[cycle][phase] = sample & 0b0000001111111111; //save sample
120       }
121       start_DAC_flag = FALSE; //STOP DAC OUTPUT
122       TU2_Deinit(TU2Ptr);//TU2 deinitialisation
123       LED1_Neg();//red led on for user frequency step finished feedback
124       //FLASH transfer
125       for(j = 0; j < MEASURE_CYCLES; j++){
126         for(i = 0; i < LUT_LENGTH; i++){
127           IFsh1_SetWordFlash(0xA410+(2*flash_index), (word)data[j][i]);
128           flash_index++;
129         }
130       }
131       IFsh1_SetWordFlash(0xA410+(2*flash_index), (word)0x17);
132       flash_index++;
133       LED1_Neg();//red led off
134     }
135     //UART RESULT TRANSFER
136     LED2_Neg();//led on for user frequency finished feedback
137     for(i = 0; i < flash_index; i++){
138       IFsh1_GetWordFlash(0xA410+(2*i), temp_valuePtr);
139       fgets(input_stream, 6, stdin); //wait for 6 chars as handshake
140       datasample = temp_value;
141       printf("%d\n", datasample);
142       LED2_Neg();
143     }
144     LED2_Neg();
```

```
145       fgets(input_stream, 6, stdin);
146       printf("%d\n", key);//stop data gathering handshake
147       LED2_Neg();
148             /* omitted free memory */
149       //********END EIS ROUTINE********//
150       break;
151       /*************CA*****************/
152       case 'A':
153           //get parameters from uart
154         fgets(input_stream,5,stdin);
155         V_AMPLITUDE = atoi(input_buffer);
156         fgets(input_stream,5,stdin);
157         OFFSET = atoi(input_buffer);
158         fgets(input_stream,4,stdin);
159         time = atoi(input_buffer);
160         fgets(input_stream,4,stdin);
161         sampling_rate = atoi(input_buffer);
162         printf("\n");
163         fgets(input_stream,6,stdin);
164         clk_freq = atoi(input_buffer);
165         //set clock frequency
166               if(clk_freq > 1000)
167                 CLKADC1_SetFreqkHz((word)(clk_freq/(int)1000));
168               else if(clk_freq > 1000000)
169                 CLKADC1_SetFreqMHz((word)(clk_freq/(int)1000000));
170               else
171                 CLKADC1_SetFreqHz((word)clk_freq);
172         SPI_ADC_middle=0b1110110000000000;
173         SPI_DAC_middle=0b0000010000000000;
174         SPI_ADC_top=0b1110110000000000;
175         SPI_DAC_top=0b0000010000000000;
176         //**************SEND CFG*****************//
177         Send_Cfg(MyGPIO1Ptr, SPI_ADC_middle, SPI_DAC_middle, SPI_ADC_top ,SPI_DAC_top);
178         n_samples = sampling_rate*time;//calulate samples
179                 /* omitted memory allocation */
180         i = 0;
181         t_wait_ms = (word)((time*1000)/n_samples); //waiting time between samples (CA timing)
182         t_wait_us = (word)(1000000/clk_freq)+(1000000/(2*clk_freq)); //wait 1 cycle and a half for sampling
    at the right time
183         n_wait_flags = ((t_wait_ms*clk_freq)/(11*1000));
184         //SPI_ADC_middle 0x01
185         SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL0, 0b1);
186         SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL1, 0b0);
187         SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL2, 0b0);
188         //set voltage value
189         DAC_PDD_SetData(DAC0_BASE_PTR,(uint16_t)((OFFSET-V_AMPLITUDE)/DAC_LSB),0U);
190         for(i = 0; i<n_samples; i++){
191           sample = 1;
192           WAIT1_Waitms(t_wait_ms);
193           for(j=0;j<n_wait_flags;j++){
194             while(EOC_flag != TRUE){
195             }
196             EOC_flag = FALSE;
197           }
198           WAIT1_Waitus(t_wait_us);
199           //data from ADC
200           SM1_TransceiveBlock(SM1_DeviceData, &SPI_ADC_middle, &sample, 1);
201           data_p[i] = sample & 0b0000001111111111; //save 10 bit sample to results array
202         }
203                 /* omitted result transfer */
```

```
204                 /* omitted free memory */
205         //********END CA ROUTINE********//
206         break;
207         /***************CV**************/
208       case 'V':
209         //********START CV ROUTINE********//
210         fgets(input_buffer,5,stdin);
211         cv_min_voltage = atoi(input_buffer);
212         fgets(input_buffer,5,stdin);
213         cv_max_voltage = atoi(input_buffer);
214         fgets(input_buffer,5,stdin);
215         we_voltage = atoi(input_buffer);
216         fgets(input_stream,4,stdin);
217         scan_rate = atoi(input_buffer);
218         fgets(input_stream,8,stdin);
219         clk_freq = atoi(input_buffer);
220         fgets(input_stream,3,stdin);
221         voltage_step = atoi(input_buffer);
222         fgets(input_stream,3,stdin);
223         MEASURE_CYCLES = atoi(input_buffer);
224         //set clock frequency
225         if(clk_freq > 1000)
226           CLKADC1_SetFreqkHz((word)(clk_freq/(int)1000));
227         else if(clk_freq > 1000000)
228           CLKADC1_SetFreqMHz((word)(clk_freq/(int)1000000));
229         else
230           CLKADC1_SetFreqHz((word)clk_freq);
231         CLKADC1_SetRatio8(128);
232         SPI_ADC_middle=0b1110110000000000;
233         SPI_DAC_middle=0b0000010000000000;
234         SPI_ADC_top=0b1110110000000000;
235         SPI_DAC_top=0b0000010000000000;
236         //*************SEND CFG***************//
237         Send_Cfg(MyGPIO1Ptr, SPI_ADC_middle, SPI_DAC_middle, SPI_ADC_top ,SPI_DAC_top);
238         voltage_step = voltage_step*DAC_LSB;
239         time = (((cv_max_voltage-cv_min_voltage)*2)/scan_rate);//seconds
240         n_samples = ((cv_max_voltage-cv_min_voltage)/(float)voltage_step)*2;//number of steps for the whole
    measure in LSBs
241         final_samples = ((cv_max_voltage-we_voltage)/(float)voltage_step);
242         if((n_samples/time)>(clk_freq/11)){
243           printf("Scan rate too high\n");
244           break;
245           LED1_Neg();
246           LED2_Neg();
247         }
248         //        scan_rate = (((cv_max_voltage-cv_min_voltage)*2)/time);
249         //        n_samples = time*sampling_rate;
250         //        single_slope_steps = sampling_rate*0.5;
251         //        voltage_step = ((cv_max_voltage-cv_min_voltage)/DAC_LSB)/single_slope_steps; //in LSBs
252         //Set initial voltage before pluggin in the sensor
253         V_AMPLITUDE = (we_voltage)/DAC_LSB; //convert mV to LSBs
254         DAC_PDD_SetData(DAC0_BASE_PTR,(uint16_t)(V_AMPLITUDE),0U);
255                 /* omitted memory allocation */
256         for(k= 0; k<MEASURE_CYCLES; k++){
257           //SPI_ADC_middle 0x01
258           SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL0, 0b1);
259           SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL1, 0b0);
260           SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL2, 0b0);
261           for(i = 0; i < n_samples; i++){
262             sample = 0;//default sample value for debug
```

```
263             if(i == n_samples || i == 0){ //sampling_rate samples per measurement
264               V_AMPLITUDE = we_voltage/DAC_LSB;
265             }
266             else if(i < (n_samples/2)-final_samples){
267               V_AMPLITUDE = V_AMPLITUDE - voltage_step; //decrease
268             }
269             else if(i >= ((n_samples/2)-final_samples) && i < (n_samples-final_samples)){
270               V_AMPLITUDE = V_AMPLITUDE + voltage_step; //increase
271             }
272             else if(i >= (n_samples-final_samples)){
273               V_AMPLITUDE = V_AMPLITUDE - voltage_step; //decrease
274             }
275             DAC_PDD_SetData(DAC0_BASE_PTR,(uint16_t)(V_AMPLITUDE),0U);
276             while(EOC_flag != TRUE){
277             }
278             EOC_flag = FALSE;
279             //data from ADC
280             SM1_TransceiveBlock(SM1_DeviceData, &SPI_ADC_middle, &sample, 1);
281             data_p[i] = sample & 0b0000001111111111; //save 10 bit sample to results array
282             WAIT1_Waitms((2*time*1000)/(float)n_samples);
283           }
284                     /* omitted result transfer*/
285         }
286                     /* omitted free memory */
287         //********END CV ROUTINE********//
288         break;
289       default:
290         break;
291     }
292   }
293   //************END MAIN*************//
294   /*** conversion from sinus units to degrees ***/
295 double Get360Phase(double x){
296   double ref_phase;
297   ref_phase = (360 * x)/(double)LUT_LENGTH;
298   return ref_phase;
299 };
300 /*** SPI TRANSCEIVE ***/
301 typedef struct {
302   LDD_SPIMASTER_TError ErrFlag;         /* Error flags */
303   uint16_t InpRecvDataNum;              /* The counter of received characters */
304   uint8_t *InpDataPtr;                  /* The buffer pointer for received characters */
305   uint16_t InpDataNumReq;               /* The counter of characters to receive by ReceiveBlock() */
306   uint16_t OutSentDataNum;              /* The counter of sent characters */
307   uint8_t *OutDataPtr;                  /* The buffer pointer for data to be transmitted */
308   uint16_t OutDataNumReq;               /* The counter of characters to be send by SendBlock() */
309   LDD_TUserData *UserData;              /* User device data structure */
310 } SM1_TDeviceData;                      /* Device data structure type */
311 typedef SM1_TDeviceData* SM1_TDeviceDataPtr; /* Pointer to the device data structure */
312 /*** LOOK UP TABLE VALUE GENERATION ***/
313 uint16_t Sine_Gen(uint16_t index){
314   float phase_degrees;
315   uint16_t v_amp_digital, off_digital, lut_value;
316   phase_degrees = (index*(360))/(float)LUT_LENGTH;
317   v_amp_digital = (V_AMPLITUDE)/(float)DAC_LSB;
318   off_digital = OFFSET/(float)DAC_LSB;
319   lut_value = (v_amp_digital)*sin(phase_degrees*(M_PI/180.0f));
320   lut_value += off_digital;
321   return lut_value;
322 };
```

```
323
324
325
326 void Send_Cfg(LDD_TDeviceData * MyGPIO1Ptr, uint16 ADC_middle,uint16 DAC_middle,uint16 ADC_top,uint16 DAC_top
      ){
327   MyGPIO1Ptr = SELECT_SPI_SLAVE_Init((LDD_TUserData *)NULL);
328   //SPI_ADC_middle 0x01
329   SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL0, 0b1);          /* Configure the output value */
330   SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL1, 0b0);          /* Configure the output value */
331   SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL2, 0b0);          /* Configure the output value */
332   WAIT1_Waitms(4);
333   SM1_SendBlock(SM1_DeviceData, &ADC_middle, 1);
334   WAIT1_Waitms(4);
335   //SPI_DAC_middle 0x03
336   SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL0, 0b1);          /* Configure the output value */
337   SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL1, 0b1);          /* Configure the output value */
338   SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL2, 0b0);          /* Configure the output value */
339   WAIT1_Waitms(4);
340   SM1_SendBlock(SM1_DeviceData, &DAC_middle, 1);
341   WAIT1_Waitms(4);
342   //SPI_ADC_top 0x05
343   SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL0, 0b1);          /* Configure the output value */
344   SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL1, 0b0);          /* Configure the output value */
345   SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL2, 0b1);          /* Configure the output value */
346   WAIT1_Waitms(4);
347   SM1_SendBlock(SM1_DeviceData,  &ADC_top, 1);
348   WAIT1_Waitms(4);
349   //SPI_DAC_top 0x07
350   SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL0, 0b1);          /* Configure the output value */
351   SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL1, 0b1);          /* Configure the output value */
352   SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL2, 0b1);          /* Configure the output value */
353   WAIT1_Waitms(4);
354   SM1_SendBlock(SM1_DeviceData, &DAC_top, 1);
355   WAIT1_Waitms(4);
356   SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL0, 0b0);          /* Configure the output value */
357   SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL1, 0b0);          /* Configure the output value */
358   SELECT_SPI_SLAVE_SetFieldValue(MyGPIO1Ptr, SEL2, 0b0);          /* Configure the output value */
359 }
360
361 void Gen_LUT(void){
362   int i;
363   for(i = 0; i < LUT_LENGTH; i++){
364     LUT[i] = Sine_Gen(i);
365   }
366 }
367
368 LDD_TError SM1_TransceiveBlock(LDD_TDeviceData *DeviceDataPtr, LDD_TData *OutBufferPtr, LDD_TData *
      InBufferPtr, uint16_t Size){
369   if (((SM1_TDeviceDataPtr)DeviceDataPtr)->OutDataNumReq != 0x00U) { /* Is the previous transmit operation
        pending? */
370     return ERR_BUSY; /* If yes then error */
371   }
372   /* {Default RTOS Adapter} Critical section begin, general PE function is used */
373   EnterCritical();
374   ((SM1_TDeviceDataPtr)DeviceDataPtr)->OutDataPtr = (uint8_t*)OutBufferPtr; /* Set a pointer to the output
        data. */
375   ((SM1_TDeviceDataPtr)DeviceDataPtr)->InpDataPtr = (uint8_t*)InBufferPtr; /* Set a pointer to the output
        data. */
376   ((SM1_TDeviceDataPtr)DeviceDataPtr)->OutDataNumReq = Size; /* Set the counter of characters to be sent. */
377   ((SM1_TDeviceDataPtr)DeviceDataPtr)->InpDataNumReq = Size; /* Set the counter of characters to be sent. */
```

```
  ((SM1_TDeviceDataPtr)DeviceDataPtr)->OutSentDataNum = 0x00U; /* Clear the counter of sent characters. */
  ((SM1_TDeviceDataPtr)DeviceDataPtr)->InpRecvDataNum = 0x00U; /* Set number of received characters to zero. */
  SPI_PDD_EnableInterruptMask(SPI0_BASE_PTR, SPI_PDD_TX_BUFFER_EMPTY); /* Enable Tx buffer empty interrupt */
  SPI_PDD_EnableInterruptMask(SPI0_BASE_PTR, SPI_PDD_RX_BUFFER_FULL_OR_FAULT); /* Enable Rx buffer full interrupt */
  /* {Default RTOS Adapter} Critical section end, general PE function is used */
  ExitCritical();
  return ERR_OK; /* OK */
}
```

A.1.2 Matlab

```
%%Clear all variables
clc;
clear all;
%SerialPort='/dev/tty.usbserial-FTWQLPCT'; %serial port mac
SerialPort='com20'; %serial port pc
TimeInterval=0.000001;%time interval between each input.
%%Set up the serial port object
s = serial(SerialPort);
set(s, 'InputBufferSize', 8); %number of bytes in inout buffer
set(s, 'OutputBufferSize', 16); %number of bytes in inout buffer
set(s, 'BaudRate', 115200);
set(s, 'Timeout', 10);
set(s, 'Terminator', 'LF');
fopen(s);
frequency = 0;
y_reference = 0;
%% DEBUG CONSOLE OUTPUT
input('Reset controller and press enter to start... ', 's');
%% GUI FOR MEASUREMENT INPUT VALUES
prompt = {'Measurement type (IS, CV, CA):'};
title = 'Measurement type dialog';
dims = [1 35];
definput = {'CV'};
parameters = inputdlg(prompt,title,dims,definput);
meas_type = parameters{:};
switch meas_type
    case 'IS'
        %DIALOG WINDOW
        prompt = {'Amplitude (mVpp):', 'offset(mV)', 'sine cycles(1-8)','number of frequency points','use internal adc? (y/n)','clk frequency (Hz)', 'gain (1-15)', 'initial waiting time (seconds)'};
        title = 'IS parameters input';
        dims = [1 35];
        definput = {'100','1650','3','9','n','50000','1','10'};
        IS_parameters = inputdlg(prompt,title,dims,definput);
        IS_input = char(IS_parameters);
        %PARSE DIALOG DATA
        Vin_peak = (str2num(IS_input(1,:))/2)/1000;
        v_cm = (str2num(IS_input(2,:))/1000);
        cycles = str2num(IS_input(3,:));
        frequency = zeros(1, str2num(IS_input(4,:)));
        freq_steps = str2num(IS_input(4,:));
        ADC_trigger = IS_input(5,1);
        f = str2num(IS_input(6,:));
        G = str2num(IS_input(7,:));
        wait_time = str2num(IS_input(8,:));
```

```
        for i = 1:freq_steps
          prompt = {'Value (mHz):'};
          title = sprintf('Frequency point %d', i);
          dims = [1 35];
          definput = {'1000'};
          freq_parameters = inputdlg(prompt,title,dims,definput);
          frequency(i) = str2num(char(freq_parameters));
         lut_length(i) = 180;
        end
        %SEND VALUES TO uC
        fprintf(s,'%s', meas_type);
        temp = num2str(Vin_peak*1000);
        send = pad(temp,4,'left','0');
        fprintf(s,'%s',send);
        temp = num2str(v_cm*1000);
        send = pad(temp,4,'left','0');
        fprintf(s,'%s',send);
        temp = num2str(cycles);
        send = pad(temp,1,'left','0');
        fprintf(s,'%s',send);
        send = sprintf('%02d', wait_time);
        fprintf(s,'%s',send);
        temp = num2str(freq_steps);
        send = pad(temp,2,'left','0');
        fprintf(s,'%s',send);
        for i=1:freq_steps
            temp = num2str(frequency(i));
            send = pad(temp,7,'left','0');
            fprintf(s,'%s',send);
        end
        temp = num2str(f);
        send = pad(temp,5,'left','0');
        fprintf(s,'%s',send);
        temp = num2str(G);
        send = pad(temp,2,'left','0');
        fprintf(s,'%s',send);
%% FIXED PARAMETERS & INITIALISATION
C = 0.000000000015;%F
ADC_LSB = 0.00322265625;
A = (2*f*C)/G;
max_lut_length = max(lut_length);
data = nan(freq_steps, max_lut_length*cycles);
amplitudes = zeros(1, freq_steps);
phases = zeros(1, freq_steps);
cyclecount = 0;
x_time(1)=1;
input('Wait for the measurement to finish and press enter...', 's');
%% CALIBRATION VALUES
phase_cal(1) = 0;
phase_cal(2) = 0;
phase_cal(3) = 0;
phase_cal(4) = 0;
phase_cal(5) = 0;
phase_cal(6) = 0;
phase_cal(7) = 0;
phase_cal(8) = 0;
phase_cal(9) = 0;
phase_cal(10) = 0;
%% GATHERING DATA
disp('Gathering data...');
```

```
for freq=1:freq_steps
  count = 1;
    while 1
    %%Serial data accessing
    send = '12345';
    fprintf(s,'%s',send);
    datasample = fscanf(s, '%d');
    if datasample == 65534
        %cyclecount = cyclecount + 1;
        break;
    elseif datasample == 23
        break;
    else
       data(freq, count) = datasample;
    end
    pause(TimeInterval);
    count = count+1;
    end
end
%% FILTER & PROCESS
disp('Filtering and processing...');
lut_steps = linspace(1,(max_lut_length*cycles), (max_lut_length*cycles));
for freq=1:freq_steps
    x = data(freq,:);
    %filtering
    [b,a] = butter(12,0.1,'low');           % IIR filter design
    y = filtfilt(b,a,x);                    % zero-phase filtering
    figure(1);
    if freq_steps>1
        subplot(freq_steps,1,freq);
    end
    plot(x); grid on ; hold on
    plot(y,'LineWidth',1.5); hold on
    clear title;
    title([num2str(frequency(freq)),' mHz']);
    %sine fitting
    [res,y_fitted, y_residual, rms_err] = sinefit(y, lut_steps, 1/max_lut_length, 0,0);
    plot(y_fitted);
    legend('ADC Data','Filtered Data','Fit Data');
    amplitudes(freq) = (res(2))*ADC_LSB; %peak voltage at integrator output
    phases(freq) = res(4);  %phase in radiants
end
for i = 1:freq_steps
    %change phase units
    phases_degrees(i) = ((phases(i)*180)/pi)-90+phase_cal(i); %phase in degrees + shift to 0 degrees
     %change amplitude units
    amplitudes_Ampere_peak(i) = amplitudes(i)*A; %peak current through impedance
    %convert frequency to Hz for plot
    frequency(i) = frequency(i)/1000;
    R(i) = abs(Vin_peak/amplitudes_Ampere_peak(i));
    %impedance calculation
    Z_mod(i) = R(i)*acos(deg2rad(phases_degrees(i)));
    Z_phase(i) =  phases_degrees(i);
    %real and imaginary part calculation
    Z_real(i) = R(i);
    Z_img(i) = Z_mod(i)*sin(deg2rad(phases_degrees(i)));
end
m = linspace(0, 2*pi, max_lut_length);
%% PLOTS
%bode
```

```
figure(2);
subplot(2,1,1);
set(gca, 'XScale', 'log');
set(gca, 'YScale', 'log');
clear title;
title('Amplitude');
xlabel('frequency [Hz]'); ylabel('amplitude [\Omega]');
grid on ; hold on;
scatter(frequency, Z_mod);
subplot(2,1,2);
set(gca, 'XScale', 'log');
clear title;
title('Phase');
xlabel('frequency [Hz]'); ylabel('phase [degrees]')
grid on ; hold on;
scatter(frequency, Z_phase);
%nyquist
figure(3);
set(gca, 'XScale', 'linear');
clear title;
title('Nyquist');xlabel('Z_{real} [\Omega]'); ylabel('Z_{img} [\Omega]')
grid on ; hold on;
scatter(Z_real, Z_img);
    case 'CV'
        %DIALOG WINDOW
        prompt = {'minimum CV voltage (mV)', 'maximum CV voltage (mV)', 'we voltage (mV)', 'scan rate (mV/s)','clk frequency (
     Hz)', 'step (LSBs = 0.8mV)', '# of cycles'};
        title = 'CV parameters input';
        dims = [1 35];
        definput = {'1520','1720','1620','100','100000', '3', '1'};
        CV_parameters = inputdlg(prompt,title,dims,definput);
        CV_input = char(CV_parameters);

        %PARSE DIALOG DATA
        min_voltage = str2num(CV_input(1,:));
        max_voltage = str2num(CV_input(2,:));
        we_voltage = str2num(CV_input(3,:));
        scan_rate = str2num(CV_input(4,:));
        f = str2num(CV_input(5,:));
        step = str2num(CV_input(6,:));
        cycles = str2num(CV_input(7,:));
        %SEND VALUES TO uC
        fprintf(s,'%s', meas_type);
        send = sprintf('%04d', min_voltage);
        fprintf(s,'%s',send);
        send = sprintf('%04d', max_voltage);
        fprintf(s,'%s',send);
        send = sprintf('%04d', we_voltage);
        fprintf(s,'%s',send);
        send = sprintf('%03d', scan_rate);
        fprintf(s,'%s',send);
        send = sprintf('%07d', f);
        fprintf(s,'%s',send);
        send = sprintf('%02d', step);
        fprintf(s,'%s',send);
        send = sprintf('%02d', cycles);
        fprintf(s,'%s',send);
    %% Set up the figure
    I = 0;
figureHandle = figure('NumberTitle','off',...
```

```
    'Name','CV Data Stream',...
    'Color',[1 1 1],'Visible','off');
% Set axes
axesHandle = axes('Parent',figureHandle,...
    'YGrid','on',...
    'YColor',[0 0 1],...
    'XGrid','on',...
    'XColor',[0 0 1],...
    'Color',[1 1 1]);
hold on;
plotHandle = plot(axesHandle,I,'Marker','.','LineWidth',1,'Color',[1 0 0]);
xlabel('Voltage [mV]','FontWeight','bold','FontSize',14,'Color',[0 0 1]);
ylabel('Current [A]','FontWeight','bold','FontSize',14,'Color',[0 0 1]);
clear title;
title('CV Data Stream','FontSize',15,'Color',[0 0 0]);
% Output file
outputfile = ['cv_results_h',datestr(now, 'HHMM_ddmmm'), '.xlsx'];
sheet = 'ResultVectors';
%% DEFAULT VALUES
v_cm = 1.65;%V
G = 1;
C = 0.000000000015;%F
A = (2*f*C*5)/G;
re_offset = 0; %mV
ADC_LSB = 0.00322265625; %V
DAC_LSB = 0.8056640625; %mV
%% Initializing variables
volt_step = floor(step * DAC_LSB);
time = (max_voltage-min_voltage)/scan_rate;%s
slope_steps = floor((max_voltage-min_voltage)/(volt_step));
min = -(max_voltage + re_offset - we_voltage);
final_steps = (max_voltage-we_voltage)/volt_step;
data = NaN(cycles, (slope_steps*2));
avg_data = NaN(cycles, (slope_steps*2));
for i=1:(slope_steps+1)
    ref_data(i) = min +  ((i*volt_step)-volt_step);
end
I(1)=1;
voltage(1)=1;
count = 1;
for cycle_index = 1:cycles
 input('Wait for the measurement to finish and press enter...', 's');
 %% GATHERING DATA
while 1
    send = '12345';
    fprintf(s,'%s',send);
    datasample = fscanf(s, '%d');
    if datasample == 23
        count = 1;
        break;
    else
        I(count) = datasample;
    end
    I(1) = 0;
    %reference voltage array generation
    if count < (slope_steps-final_steps)
        voltage(count) = ref_data(final_steps+count);
    elseif count >= (slope_steps-final_steps) && count <= ((slope_steps*2)-final_steps)
        voltage(count) = ref_data(((slope_steps*2)+1)-(count+final_steps));
    elseif count > ((slope_steps*2)-final_steps) && count  <= (slope_steps*2)
```

```
        voltage(count) = ref_data(count-((slope_steps*2)-final_steps));
    end
    if datasample ~= 23
    I(count) = (((I(count)*ADC_LSB)-v_cm)*A);
    end
    set(plotHandle,'XData',voltage);
    set(plotHandle,'YData',I);
    set(figureHandle,'Visible','on');
    pause(TimeInterval);
    count = count+1;
end
data(cycle_index, :) = I(:);
xlrange = ['A',num2str(cycle_index)];
xlswrite(outputfile, I, sheet, xlrange);
end
figure(3); hold on
title('CV Data - Mooncake');
for i = 5:cycles
   plot(voltage,data(i,:));
end
figure(4); hold on
title('CV Data - Mooncake - Post-Processed');
for i = 1:cycles
   avg_data(i, :) = movmean(data(i,:),20);
end
for i = 5:cycles
   plot(voltage,avg_data(i,:));
end
legend('0mM','1mM','2mM','3mM','4mM','5mM','6mM','7mM');
    case 'CA'
    %case CA omitted
end
%% Clean up the serial port
fclose(s);
delete(s);
delete(instrfindall);
clear s;
```

A.2 PCB Firmware

A.2.1 Chronoamperometry

```
//This code is an adaptation of Amperometric.c example
//{Copyright (c) 2014 Analog Devices}
#include <stdio.h>
#include <string.h>
#include "test_common.h"
#include "afe.h"
#include "afe_lib.h"
#include "uart.h"

/* Macro to enable the returning of AFE data using the UART */
/*       1 = return AFE data on UART                        */
/*       0 = return AFE data on SW (Std Output)             */
#define USE_UART_FOR_DATA           (1)
```

```
14 /* DC Level 1 voltage in mV (range: -0.8V to 0.8V) */
15 #define VL1                          (-600)
16 /* DC Level 2 voltage in mV (range: -0.8V to 0.8V) */
17 #define VL2                          (-600)
18 /* The duration (in us) of DC Level 1 voltage */
19 #define DURL1                        ((uint32_t)(60000000)) //50sec
20 /* The duration (in us) of DC Level 2 voltage */
21 #define DURL2                        ((uint32_t)(60000000)) //0sec
22 /* The duration (in us) which the IVS switch should remain closed (to shunt */
23 /* switching current) before changing the DC level.                         */
24 #define DURIVS1                      ((uint32_t)(1))
25 /* The duration (in us) which the IVS switch should remain closed (to shunt */
26 /* switching current) after changing the DC level.                          */
27 #define DURIVS2                      ((uint32_t)(1))
28 /* Is shunting of the switching current required? Required: 1, Not required: 0 */
29 #define SHUNTREQD                    (0)
30 /* RCAL value, in ohms                                                  */
31 /* Default value on ADuCM350 Switch Mux Config Board Rev.0 is 1k    */
32 #define RCAL                         (10000)
33 /* RTIA value, in ohms                                                  */
34 /* Default value on ADuCM350 Switch Mux Config Board Rev.0 is 7.5k  */
35 #define RTIA                         (1000000)
36 /* DO NOT EDIT: DAC LSB size in mV, before attenuator (1.6V / (2^12 - 1)) */
37 #define DAC_LSB_SIZE                 (0.39072)
38 /* DO NOT EDIT: DC Level 1 in DAC codes */
39 #define DACL1                        ((uint32_t)(((float)VL1 / (float)DAC_LSB_SIZE) + 0x800))
40 /* DO NOT EDIT: DC Level 2 in DAC codes */
41 #define DACL2                        ((uint32_t)(((float)VL2 / (float)DAC_LSB_SIZE) + 0x800))
42 /* DO NOT EDIT: Number of samples to be transferred by DMA, based on the duration of the sequence.*/
43 /* SAMPLE_COUNT = (Level 1 Duration + Level 2 Duration)us * (160k/178)samples/s      */
44 #define SAMPLE_COUNT                 (uint32_t)((2 * (DURL1 + DURL2)) / 2225)
45 /* Size limit for each DMA transfer (max 1024) */
46 #define DMA_BUFFER_SIZE              (1000u)
47 /* DO NOT EDIT: Maximum printed message length. Used for printing only. */
48 #define MSG_MAXLEN                   (50)
49 #pragma location="volatile_ram"
50 uint16_t        dmaBuffer[DMA_BUFFER_SIZE * 2];
51 /* Sequence for Amperometric measurement */
52 uint32_t seq_afe_ampmeas[] = {
53     0x00150065,/*  0 - Safety Word, Command Count = 15, CRC = 0x1C*/
54     0x84007818,/*  1 - AFE_FIFO_CFG: DATA_FIFO_SOURCE_SEL = 0b11 (LPF)*/
55     0x8A000030,/*  2 - AFE_WG_CFG: TYPE_SEL = 0b00*/
56     0x88000F00,/*  3 - AFE_DAC_CFG: DAC_ATTEN_EN = 0 (disable DAC attenuator)*/
57     0xAA000800,/*  4 - AFE_WG_DAC_CODE: DAC_CODE = 0x800 (DAC Level 1 placeholder, user programmable)*/
58     0xA0000002/*  5 - AFE_ADC_CFG: MUX_SEL = 0b00010, GAIN_OFFS_SEL = 0b00 (TIA)*/
59     0xA2000000,/*  6 - AFE_SUPPLY_LPF_CFG: BYPASS_SUPPLY_LPF = 0 (do not bypass)*/
60     0x86003342,/*  7 - DMUX_STATE = 2, PMUX_STATE = 4, NMUX_STATE = 3, TMUX_STATE = 3 */
61     0x0001A900,/*  8 - Wait: 6.8ms (based on load RC = 6.8kOhm * 1uF)*/
62     0x80024EF0,/*  9 - AFE_CFG: WG_EN = 1*/
63     0x00000C80,/* 10 - Wait: 200us*/
64     0x80034FF0,/* 11 - AFE_CFG: ADC_CONV_EN = 1, SUPPLY_LPF_EN = 1*/
65     0x00090880,/* 12 - Wait: 37ms  for LPF settling*/
66     0x00000000,/* 13 - Wait: (DAC Level 1 duration - IVS duration 1) (placeholder, user programmable) */
67     0x86013322,/* 14 - IVS_STATE = 1 (close IVS switch, user programmable)*/
68     0x00000000,/* 15 - Wait: IVS duration 1 (placeholder, user programmable)*/
69     0xAA000800,/* 16 - AFE_WG_DAC_CODE: DAC_CODE = 0x800 (DAC Level 2 placeholder, user programmable)*/
70     0x00000000,/* 17 - Wait: IVS duration 2 (placeholder, user programmable)*/
71     0x86003322,/* 18 - IVS_STATE = 0 (open IVS switch)*/
72     0x00000000,/* 19 - Wait: (DAC Level 2 duration - IVS duration 2) (placeholder, user programmable)*/
73     0x80020EF0,/* 20 - AFE_CFG: WAVEGEN_EN = 0, ADC_CONV_EN = 0, SUPPLY_LPF_EN = 0*/
```

```
    0x82000002,/* 21 - AFE_SEQ_CFG: SEQ_EN = 0 */
};
/* Variables and functions needed for data output through UART */
ADI_UART_HANDLE     hUartDevice     = NULL;
/* Function prototypes */
void                    test_print                  (char *pBuffer);
ADI_UART_RESULT_TYPE    uart_Init                   (void);
ADI_UART_RESULT_TYPE    uart_UnInit                 (void);
extern int32_t          adi_initpinmux              (void);
void        RxDmaCB         (void *hAfeDevice,
                             uint32_t length,
                             void *pBuffer);
void printString(char *ptr);
int main(void) {
    ADI_AFE_DEV_HANDLE  hAfeDevice;
    uint32_t            dur1;
    uint32_t            dur2;
    uint32_t            dur3;
    uint32_t            dur4;

    ADI_UART_GENERIC_SETTINGS_TYPE  UART_Settings;
    UART_Settings.BaudRate = ADI_UART_BAUD_57600;
    UART_Settings.bBlockingMode = true;
    UART_Settings.bInterruptMode = false;
    UART_Settings.Parity = ADI_UART_PARITY_NONE;
    UART_Settings.WordLength = ADI_UART_WLS_8;
    UART_Settings.bDmaMode = false;
    /* Initialize system */
    SystemInit();
    /* Change the system clock source to HFXTAL and change clock frequency to 16MHz     */
    /* Requirement for AFE (ACLK)                                                        */
    SystemTransitionClocks(ADI_SYS_CLOCK_TRIGGER_MEASUREMENT_ON);
    /* SPLL with 32MHz used, need to divide by 2 */
    SetSystemClockDivider(ADI_SYS_CLOCK_UART, 2);
     /*******************************INITIALISATION*****************************/
    /* Test initialization */
    test_Init();
    /* initialize static pinmuxing */
    adi_initpinmux();
     /* Initialize the UART for transferring measurement data out */
    adi_UART_Init(ADI_UART_DEVID_0, &hUartDevice, NULL);
    /*Set UART Communication Parameters*/
    adi_UART_SetGenericSettings(hUartDevice, &UART_Settings);
    adi_UART_Enable(hUartDevice, true);
    /* Initialize GPIO */
    if (ADI_GPIO_SUCCESS != adi_GPIO_Init())
    {
        printString("adi_GPIO_Init");//***debug feedback
    }
    /*Enable and Set GPIO Pins for RCAL and RTIA*/
    //RTIA Enable
    adi_GPIO_SetOutputEnable(ADI_GPIO_PORT_1, ADI_GPIO_PIN_0, true);
    adi_GPIO_SetOutputEnable(ADI_GPIO_PORT_1, ADI_GPIO_PIN_1, true);
    adi_GPIO_SetOutputEnable(ADI_GPIO_PORT_1, ADI_GPIO_PIN_2, true);
    adi_GPIO_SetOutputEnable(ADI_GPIO_PORT_1, ADI_GPIO_PIN_3, true);
    //RCAL Enable
    adi_GPIO_SetOutputEnable(ADI_GPIO_PORT_1, ADI_GPIO_PIN_4, true);
    adi_GPIO_SetOutputEnable(ADI_GPIO_PORT_1, ADI_GPIO_PIN_5, true);
    adi_GPIO_SetOutputEnable(ADI_GPIO_PORT_1, ADI_GPIO_PIN_6, true);
    adi_GPIO_SetOutputEnable(ADI_GPIO_PORT_1, ADI_GPIO_PIN_7, true);
```

```
134     //RCAL Set (PIN_4 = 100R, PIN_5 = 10K, PIN_6 = 10M, PIN_7 = 100M)
135     adi_GPIO_SetHigh(ADI_GPIO_PORT_1, ADI_GPIO_PIN_5);//HIGH
136     adi_GPIO_SetLow(ADI_GPIO_PORT_1, ADI_GPIO_PIN_4);
137     adi_GPIO_SetLow(ADI_GPIO_PORT_1, ADI_GPIO_PIN_6);
138     adi_GPIO_SetLow(ADI_GPIO_PORT_1, ADI_GPIO_PIN_7);
139     //RTIA Set (PIN_1 = 100R, PIN_2 = 10K, PIN_3 = 10M, PIN_0 = 100M)
140     adi_GPIO_SetHigh(ADI_GPIO_PORT_1, ADI_GPIO_PIN_2);//HIGH
141     adi_GPIO_SetLow(ADI_GPIO_PORT_1, ADI_GPIO_PIN_0);
142     adi_GPIO_SetLow(ADI_GPIO_PORT_1, ADI_GPIO_PIN_1);
143     adi_GPIO_SetLow(ADI_GPIO_PORT_1, ADI_GPIO_PIN_3);
144     /* Initialize the AFE API */
145     if (ADI_AFE_SUCCESS != adi_AFE_Init(&hAfeDevice))
146     {
147         printString("Init");
148     }
149     /* Set RCAL Value */
150     if (ADI_AFE_SUCCESS != adi_AFE_SetRcal(hAfeDevice, RCAL))
151     {
152         printString("Set RCAL");
153     }
154     /* Set RTIA Value */
155     if (ADI_AFE_SUCCESS != adi_AFE_SetRtia(hAfeDevice, RTIA))
156     {
157         printString("Set RTIA");
158     }
159     /* AFE power up */
160     if (ADI_AFE_SUCCESS != adi_AFE_PowerUp(hAfeDevice))
161     {
162         printString("PowerUp");
163     }
164     /* Excitation Channel Power-Up */
165     if (ADI_AFE_SUCCESS != adi_AFE_ExciteChanPowerUp(hAfeDevice))
166     {
167         printString("ExciteChanCalAtten");
168     }
169     /* TIA Channel Calibration */
170     if (ADI_AFE_SUCCESS != adi_AFE_TiaChanCal(hAfeDevice))
171     {
172         printString("TiaChanCal");
173     }
174     /* Excitation Channel (no attenuation) Calibration */
175     if (ADI_AFE_SUCCESS != adi_AFE_ExciteChanCalNoAtten(hAfeDevice))
176     {
177             printString("adi_AFE_ExciteChanCalNoAtten");
178         }
179     /**********END INITIALISATION**************/
180
181     /**************MEASUREMENT*****************/
182     /* Set the user programmable portions of the sequence */
183     /* Set the duration values */
184     if (SHUNTREQD)
185     {
186     dur1 = DURL1 - DURIVS1;
187     dur2 = DURIVS1;
188     dur3 = DURIVS2;
189     dur4 = DURL2 - DURIVS2;
190     }
191     else
192     {
193         dur1 = DURL1;
```

```
194         dur2 = 0;
195         dur3 = 0;
196         dur4 = DURL2;
197     }
198     /* Set durations in ACLK periods */
199         seq_afe_ampmeas[13] = dur1 * 16;
200         seq_afe_ampmeas[15] = dur2 * 16;
201         seq_afe_ampmeas[17] = dur3 * 16;
202         seq_afe_ampmeas[19] = dur4 * 16;
203     /* Set DAC Level 1 */
204     seq_afe_ampmeas[4]  = SEQ_MMR_WRITE(REG_AFE_AFE_WG_DAC_CODE, DACL1);
205     /* Set DAC Level 2 */
206     seq_afe_ampmeas[16] = SEQ_MMR_WRITE(REG_AFE_AFE_WG_DAC_CODE, DACL2);
207     if (!SHUNTREQD)
208     {
209         /* IVS switch remains open */
210         seq_afe_ampmeas[14] &= 0xFFFEFFFF;
211     }
212 #if (ADI_AFE_CFG_ENABLE_RX_DMA_DUAL_BUFFER_SUPPORT == 1)
213     /* Set the Rx DMA buffer sizes */
214     if (ADI_AFE_SUCCESS != adi_AFE_SetDmaRxBufferMaxSize(hAfeDevice, DMA_BUFFER_SIZE, DMA_BUFFER_SIZE))
215     {
216         printString("adi_AFE_SetDmaRxBufferMaxSize");
217     }
218 #endif /* ADI_AFE_CFG_ENABLE_RX_DMA_DUAL_BUFFER_SUPPORT == 1 */
219
220     /* Register Rx DMA Callback */
221     if (ADI_AFE_SUCCESS != adi_AFE_RegisterCallbackOnReceiveDMA(hAfeDevice, RxDmaCB, 0))
222     {
223         printString("adi_AFE_RegisterCallbackOnReceiveDMA");
224     }
225     /* Recalculate CRC in software for the amperometric measurement */
226     adi_AFE_EnableSoftwareCRC(hAfeDevice, true);
227     //printString("===>MEASUREMENT STARTS<===");
228     /* Perform the Amperometric measurement(s) */
229     if (ADI_AFE_SUCCESS != adi_AFE_RunSequence(hAfeDevice, seq_afe_ampmeas, (uint16_t *) dmaBuffer,
        SAMPLE_COUNT))
230     {
231         printString("adi_AFE_RunSequence");
232     }
233     //printString("===>MEASUREMENT ENDS<===");
234     /* Restore to using default CRC stored with the sequence */
235     adi_AFE_EnableSoftwareCRC(hAfeDevice, false);
236     /* AFE Power Down */
237     if (ADI_AFE_SUCCESS != adi_AFE_PowerDown(hAfeDevice))
238     {
239         printString("adi_AFE_PowerDown");
240     }
241     /* Unregister Rx DMA Callback */
242     if (ADI_AFE_SUCCESS != adi_AFE_RegisterCallbackOnReceiveDMA(hAfeDevice, NULL, 0))
243         {
244         printString("adi_AFE_RegisterCallbackOnReceiveDMA (unregister)");
245         }
246     /* Uninitialize the AFE API */
247     if (ADI_AFE_SUCCESS != adi_AFE_UnInit(hAfeDevice))
248     {
249         printString("adi_AFE_UnInit");
250     }
251     printString("23");
252     //printString("Chronoamperometry finished!");
```

```
}
void RxDmaCB(void *hAfeDevice, uint32_t length, void *pBuffer)
{
    char                    msg[MSG_MAXLEN];
    uint32_t                i;
    uint16_t                *ppBuffer = (uint16_t*)pBuffer;
    /* Check if there are samples to be sent */
    if (length)
    {
        for (i = 0; i < length; i++)
        {
            sprintf(msg, "%u", *ppBuffer++);
            printString(msg);
    }
    }
}
/*print through UART*/
 void printString(char *ptr){
    char msg[MSG_MAXLEN];
    sprintf(msg, "%s\n", ptr);
    int16_t size = strlen(msg);
    adi_UART_BufTx(hUartDevice, msg, &size);
}
```

A.2.2 Impedance Spectroscopy

```
//This code is an adaptation of ImpedanceMeasurement_2Wire.c example
//{Copyright (c) 2014 Analog Devices}
//Added capabilities: NFC com, UART debug, 3Wire cfg, Spectroscopy sweep, GPIO switches
volatile    bool_t              bCommonInterruptFlag;
int SystickInter = 0;
int Count = 0;
/*Start Frequency in Hz */
#define START_FREQ  (1000)
/*Stop Frequency in Hz */
#define STOP_FREQ   (2000)
/*# frequency points*/
#define N_POINTS (2)
/* Peak voltage in mV */
#define VPEAK                       (10)
/* RCAL value, in ohms */
#define RCAL                        (10000)
/* RTIA value, in ohms                                          */
#define RTIA                        (1000000)
/* DAC LSB size in mV, before attenuator (1.6V / (2^12 - 1)) */
#define DAC_LSB_SIZE                (0.39072)
/* Sine amplitude in DAC codes */
#define SINE_AMPLITUDE              ((uint16_t)((VPEAK) / DAC_LSB_SIZE)) //NO ATTENUATOR
//#define SINE_AMPLITUDE              ((uint16_t)((VPEAK * 40) / DAC_LSB_SIZE))
/* If both real and imaginary result are within the interval (DFT_RESULTS_OPEN_MIN_THR,
    DFT_RESULTS_OPEN_MAX_THR), */
/* it is considered an open circuit and results for both magnitude and phase will be 0*/
#define DFT_RESULTS_OPEN_MAX_THR    (1)
#define DFT_RESULTS_OPEN_MIN_THR    (-1)
/* The number of results expected from the DFT, e.g. 8 for 4 complex results */
#define DFT_RESULTS_COUNT           (4)
/* Fractional LSB size for the fixed32_t type defined below, used for printing only. */
#define FIXED32_LSB_SIZE            (625)
```

```
#define MSG_MAXLEN                  (50)
/* SPI device Info */
#define SPI_DEVICE_NUM          ADI_SPI_DEVID_H
#define SPI_CS_PORT             ADI_GPIO_PORT_0
#define SPI_CS_PIN              ADI_GPIO_PIN_15
#define SPI_CLOCK               1000000
/* Buffer sizes */
#define PROLOGUE_SIZE           8u
#define TRANSFER_SIZE           1024u
#define CHIPID_SIZE             2u
#define EEPROM_BASE_ADDR        0x04
bool NFC_SPI_Write_EEPROM(uint8_t address, uint8_t data0, uint8_t data1, uint8_t data2, uint8_t data3);
/*Global Variables*/
char                msg[MSG_MAXLEN];
/* Custom fixed-point type used for final results,              */
/* to keep track of the decimal point position.                 */
/* Signed number with 28 integer bits and 4 fractional bits.    */
typedef union {
    int32_t     full;
    struct {
        uint8_t fpart:4;
        int32_t ipart:28;
    } parts;
} fixed32_t;
/*global data buffer*/
uint8_t gTransmitBuffer[TRANSFER_SIZE];
uint8_t gPrologueBuffer[PROLOGUE_SIZE];
uint8_t gGarbageBuffer[TRANSFER_SIZE];
/*Device declaration*/
ADI_UART_HANDLE     hUartDevice     = NULL; //UART
ADI_SPI_DEV_HANDLE hSPIDevice = NULL;//SPI
/* Function prototypes */
q15_t           arctan                  (q15_t imag, q15_t real);
fixed32_t       calculate_magnitude     (q31_t magnitude_rcal, q31_t magnitude_z);
fixed32_t       calculate_phase         (q15_t phase_rcal, q15_t phase_z);
void            convert_dft_results     (int16_t *dft_results, q15_t *dft_results_q15, q31_t *dft_results_q31
    );
void                    sprintf_fixed32         (char *out, fixed32_t in);
void                    print_MagnitudePhase    (char *text, fixed32_t magnitude, fixed32_t phase);
extern int32_t          adi_initpinmux          (void);
void            printString(char *ptr);
void            ConfigureSPI(ADI_SPI_DEV_HANDLE hSpi);
void            NFC_SPI_Write_EEPROM(uint8_t address, uint8_t data0, uint8_t data1, uint8_t data2, uint8_t
    data3);
/* Sequence for AC measurement, performs 4 DFTs:        */
/*     RCAL, AFE3-AFE4, AFE4-AFE5, AFE3-AFE5            */
uint32_t seq_afe_acmeas3wire[] = {
    0x00130043,   /*0* Safety word: bits 31:16 = command count, bits 7:0 = CRC */
    0x84005818,   /*1* AFE_FIFO_CFG: DATA_FIFO_SOURCE_SEL = 10 */
    0x8A000034,   /*2* AFE_WG_CFG: TYPE_SEL = 10 */
    0x98000000,   /*3* AFE_WG_CFG: SINE_FCW = 0 (placeholder, user programmable) */
    0x9E000000,   /*4* AFE_WG_AMPLITUDE: SINE_AMPLITUDE = 0 (placeholder, user programmable) */
    0x88000F00,   /*5* AFE_DAC_CFG: DAC_ATTEN_EN = 0 */
    0xA0000002,   /*6* AFE_ADC_CFG: MUX_SEL = 00010, GAIN_OFFS_SEL = 00 */
    /* RCAL */
    0x86008811,   /*7* DMUX_STATE = 1, PMUX_STATE = 1, NMUX_STATE = 8, TMUX_STATE = 8 */
    0x00000640,   /*8* Wait 100us */
    0x80024EF0,   /*9* AFE_CFG: WAVEGEN_EN = 1 */
    0x00000C80,   /*10* Wait 200us */
    0x8002CFF0,   /*11* AFE_CFG: ADC_CONV_EN = 1, DFT_EN = 1 */
```

```
    0x00032340,    /*12* Wait 13ms */
    0x80024EF0,    /*13* AFE_CFG: ADC_CONV_EN = 0, DFT_EN = 0 */
    /* AFE2 - AFE3 - AFE4 */
    0x86003342,    /*14* DMUX_STATE = 2, PMUX_STATE = 4, NMUX_STATE = 3, TMUX_STATE = 3 */
    0x00000640,    /*15* Wait 100us */
    0x8002CFF0,    /*16* AFE_CFG: ADC_CONV_EN = 1, DFT_EN = 1 */
    0x00032340,    /*17* Wait 13ms */
    0x80024EF0,    /*18* AFE_CFG: ADC_CONV_EN = 0, DFT_EN = 0 */
    0x82000002,    /*19* AFE_SEQ_CFG: SEQ_EN = 0 */
};
int main(void) {
    ADI_AFE_DEV_HANDLE  hAfeDevice;
    int16_t             dft_results[DFT_RESULTS_COUNT+1];
    q15_t               dft_results_q15[DFT_RESULTS_COUNT+1];
    q31_t               dft_results_q31[DFT_RESULTS_COUNT+1];
    q31_t               magnitude[DFT_RESULTS_COUNT / 2];
    q15_t               phase[DFT_RESULTS_COUNT / 2];
    fixed32_t           magnitude_result[DFT_RESULTS_COUNT / 2 - 1];
    fixed32_t           phase_result[DFT_RESULTS_COUNT / 2 - 1];
    uint32_t            frequency[N_POINTS];
    uint32_t            fcw;
    uint16_t            i,freq_step;
    uint8_t              freq_index;
    uint8_t mem_addr = EEPROM_BASE_ADDR;
    uint8_t NFC_data[4];
    uint32_t temp;
    ADI_UART_GENERIC_SETTINGS_TYPE  UART_Settings;
    UART_Settings.BaudRate = ADI_UART_BAUD_9600;
    UART_Settings.bBlockingMode = true;
    UART_Settings.bInterruptMode = false;
    UART_Settings.Parity = ADI_UART_PARITY_NONE;
    UART_Settings.WordLength = ADI_UART_WLS_8;
    UART_Settings.bDmaMode = false;
/* omitted system initialization */
    /*Enable and Set GPIO Pins for RCAL and RTIA*/
    //RTIA Enable
    adi_GPIO_SetOutputEnable(ADI_GPIO_PORT_1, ADI_GPIO_PIN_0, true);
    adi_GPIO_SetOutputEnable(ADI_GPIO_PORT_1, ADI_GPIO_PIN_1, true);
    adi_GPIO_SetOutputEnable(ADI_GPIO_PORT_1, ADI_GPIO_PIN_2, true);
    adi_GPIO_SetOutputEnable(ADI_GPIO_PORT_1, ADI_GPIO_PIN_3, true);
    //RCAL Enable
    adi_GPIO_SetOutputEnable(ADI_GPIO_PORT_1, ADI_GPIO_PIN_4, true);
    adi_GPIO_SetOutputEnable(ADI_GPIO_PORT_1, ADI_GPIO_PIN_5, true);
    adi_GPIO_SetOutputEnable(ADI_GPIO_PORT_1, ADI_GPIO_PIN_6, true);
    adi_GPIO_SetOutputEnable(ADI_GPIO_PORT_1, ADI_GPIO_PIN_7, true);
    //RCAL Set (PIN_4 = 100R, PIN_5 = 10K, PIN_6 = 10M, PIN_7 = 100M)
    adi_GPIO_SetHigh(ADI_GPIO_PORT_1, ADI_GPIO_PIN_5);//HIGH SELECTED
    adi_GPIO_SetLow(ADI_GPIO_PORT_1, ADI_GPIO_PIN_4);
    adi_GPIO_SetLow(ADI_GPIO_PORT_1, ADI_GPIO_PIN_6);
    adi_GPIO_SetLow(ADI_GPIO_PORT_1, ADI_GPIO_PIN_7);
    //RTIA Set (PIN_1 = 100R, PIN_2 = 1M, PIN_3 = 10M, PIN_0 = 100M)
    adi_GPIO_SetHigh(ADI_GPIO_PORT_1, ADI_GPIO_PIN_2);//HIGH SELECTED
    adi_GPIO_SetLow(ADI_GPIO_PORT_1, ADI_GPIO_PIN_0);
    adi_GPIO_SetLow(ADI_GPIO_PORT_1, ADI_GPIO_PIN_1);
    adi_GPIO_SetLow(ADI_GPIO_PORT_1, ADI_GPIO_PIN_3);
    //SPI_CS
    adi_GPIO_SetOutputEnable(ADI_GPIO_PORT_0, ADI_GPIO_PIN_15, true);
     /* Set RCAL Value */
    if (ADI_AFE_SUCCESS != adi_AFE_SetRcal(hAfeDevice, RCAL))
    {
```

```
150         printString("Set RCAL");//***debug feedback
151     }
152     /* Set RTIA Value */
153     if (ADI_AFE_SUCCESS != adi_AFE_SetRtia(hAfeDevice, RTIA))
154     {
155         printString("Set RTIA");//***debug feedback
156     }
157     /* AFE power up */
158     if (adi_AFE_PowerUp(hAfeDevice))
159     {
160         printString("adi_AFE_PowerUp");//***debug feedback
161     }
162     /* Excitation Channel Power-Up */
163     if (adi_AFE_ExciteChanPowerUp(hAfeDevice))
164     {
165         printString("adi_AFE_ExciteChanPowerUp");//***debug feedback
166     }
167     /* TIA Channel Calibration */
168     if (adi_AFE_TiaChanCal(hAfeDevice))
169     {
170         printString("adi_AFE_TiaChanCal");//***debug feedback
171     }
172     /* Excitation Channel Calibration (Attenuation Enabled) */
173     if (adi_AFE_ExciteChanCalAtten(hAfeDevice))
174     {
175         printString("adi_AFE_ExciteChanCalAtten");//***debug feedback
176     }
177     printString("\n\n");//***debug feedback
178     /* SysTick initialization */
179     SysTick_Config(16000000/10);  /* CMSIS API */
180     fcw =((uint32_t)(((uint64_t)frequency[freq_index] << 26) / 16000000 + 0.5));
181     /* Update FCW in the sequence */
182      seq_afe_acmeas3wire[3] = SEQ_MMR_WRITE(REG_AFE_AFE_WG_FCW, fcw);
183      /* Update sine amplitude in the sequence */
184      seq_afe_acmeas3wire[4] = SEQ_MMR_WRITE(REG_AFE_AFE_WG_AMPLITUDE, SINE_AMPLITUDE);
185      /*******************Measurement Routine**************************/
186         bCommonInterruptFlag = false;
187         /* Enter CORE_SLEEP until a state chage is requested */
188         if (SystemEnterLowPowerMode(ADI_SYS_MODE_CORE_SLEEP, &bCommonInterruptFlag, 0))
189         {
190             printString("SystemEnterLowPowerMode (CORE_SLEEP) failed");
191         }
192         /*
193         while (SystickInter == 0) {
194           for (i=0; i<50; i++) {}
195         }*/
196         SystickInter = 0;
197         /* Recalculate CRC in software for the AC measurement, because we changed   */
198         /* FCW and sine amplitude settings                                          */
199         adi_AFE_EnableSoftwareCRC(hAfeDevice, true);
200         /* Perform the Impedance measurement */
201         if (adi_AFE_RunSequence(hAfeDevice, seq_afe_acmeas3wire, (uint16_t *)dft_results, DFT_RESULTS_COUNT))
202         {
203             printString("Impedance Measurement");
204         }
205         /* Restore to using default CRC stored with the sequence */
206         adi_AFE_EnableSoftwareCRC(hAfeDevice, false);
207         /* Print DFT complex results */
208         sprintf(msg, "%d Hz - DFT results :", frequency[freq_index]);
209         //printString(msg);
```

```
210         sprintf(msg, "    RCAL        = (%6d+j%6d)", dft_results[0], dft_results[1]);
211         //printString(msg);
212         sprintf(msg, "    AFE2 - AFE3 - AFE4 = (%6d+j%6d) ({Real}+j{Imaginary})", dft_results[2], dft_results
    [3]);
213         //printString(msg);
214         /* Convert DFT results to 1.15 and 1.31 formats.  */
215         convert_dft_results(dft_results, dft_results_q15, dft_results_q31);
216         /* Magnitude calculation */
217         /* Use CMSIS function */
218         arm_cmplx_mag_q31(dft_results_q31, magnitude, DFT_RESULTS_COUNT / 2);
219         /* Calculate final magnitude values, calibrated with RCAL. */
220         for (i = 0; i < DFT_RESULTS_COUNT / 2 - 1; i++)
221         {
222             magnitude_result[i] = calculate_magnitude(magnitude[0], magnitude[i + 1]);
223         }
224         /* Phase calculation */
225         /* RCAL first */
226         phase[0] = arctan(dft_results[1], dft_results[0]);
227         for (i = 0; i < DFT_RESULTS_COUNT / 2 - 1; i++)
228         {
229             /* No need to calculate the phase if magnitude is 0 (open circuit) */
230             if (magnitude_result[i].full)
231             {
232                 /* First the measured phase. */
233                 phase[i + 1]         = arctan(dft_results[2 * (i + 1) + 1], dft_results[2 * (i + 1)]);
234                 /* Then the phase calibrated with RCAL. */
235                 phase_result[i]      = calculate_phase(phase[0], phase[i + 1]);
236             }
237             else
238             {
239                 phase[i + 1]         = 0;
240                 phase_result[i].full = 0;
241             }
242         }
243         /* Print Mag and Phase */
244         sprintf(msg, "Frequency: %d Hz ", frequency[freq_index]);
245         printString(msg);
246         print_MagnitudePhase("(Magnitude, Phase)", magnitude_result[0], phase_result[0]);
247         /*Result transfer*/
248         temp = magnitude_result[0].parts.ipart;
249         NFC_data[3] = temp;
250         NFC_data[2] = temp >>  8;
251         NFC_data[1] = temp >> 16;
252         NFC_data[0] = temp >> 24;
253         sprintf(msg, "Magnitude Memory Block: %x %x %x %x", NFC_data[0], NFC_data[1], NFC_data[2], NFC_data
    [3]);
254         printString(msg);
255         NFC_SPI_Write_EEPROM(mem_addr, NFC_data[0], NFC_data[1], NFC_data[2], NFC_data[3]);//Send magnitude
    result to NFC through SPI
256         mem_addr++; //increase EEPROM address
257         for( i=0; i<0xFFFF; i++ ) ;//wait
258         temp = phase_result[0].parts.ipart;
259         NFC_data[3] = temp;
260         NFC_data[2] = temp >>  8;
261         NFC_data[1] = temp >> 16;
262         NFC_data[0] = temp >> 24;
263         sprintf(msg, "Phase Memory Block: %x %x %x %x", NFC_data[0], NFC_data[1], NFC_data[2], NFC_data[3]);
264         printString(msg);
265         NFC_SPI_Write_EEPROM(mem_addr, NFC_data[0], NFC_data[1], NFC_data[2], NFC_data[3]);//Send phase
    result to NFC through SPI
```

```
266         mem_addr++; //increase EEPROM address
267         for( i=0; i<0xFFFF; i++ ) ;//wait
268      /**************End Measurement Routine****************/
269     /* AFE Power Down */
270     if (adi_AFE_PowerDown(hAfeDevice))
271     {
272         printString("adi_AFE_PowerDown");
273     }
274     /* Uninitialize the AFE API */
275     if (adi_AFE_UnInit(hAfeDevice))
276     {
277         printString("adi_AFE_UnInit");
278     }
279     }
280     /* Uninitialize the UART */
281     adi_UART_UnInit(hUartDevice);
282     return 0;
283 }
284 /* Arctan Implementation  */
285 const q15_t coeff[5] = {
286     (q15_t)0x28BD,      /*  0.318253 */
287     (q15_t)0x006D,      /*  0.003314 */
288     (q15_t)0xEF3E,      /* -0.130908 */
289     (q15_t)0x08C6,      /*  0.068542 */
290     (q15_t)0xFED4,      /* -0.009159 */
291 };
292 q15_t arctan(q15_t imag, q15_t real) {
293     q15_t       t;
294     q15_t       out;
295     uint8_t     rotation; /* Clockwise, multiples of PI/4 */
296     int8_t      i;
297     if ((q15_t)0 == imag) {
298         /* Check the sign*/
299         if (real & (q15_t)0x8000) {
300             /* Negative, return -PI */
301             return (q15_t)0x8000;
302         }
303         else {
304             return (q15_t)0;
305         }
306     }
307     else {
308         rotation = 0;
309         /* Rotate the vector until it's placed in the first octant (0..PI/4) */
310         if (imag < 0) {
311             imag      = -imag;
312             real      = -real;
313             rotation += 4;
314         }
315         if (real <= 0) {
316             /* Using 't' as temporary storage before its normal usage */
317             t         = real;
318             real      = imag;
319             imag      = -t;
320             rotation += 2;
321         }
322         if (real <= imag) {
323             /* The addition below may overflow, drop 1 LSB precision if needed. */
324             /* The subtraction cannot underflow.                                 */
325             t = real + imag;
```

```
326             if (t < 0) {
327                 /* Overflow */
328                 t          = imag - real;
329                 real       = (q15_t)(((q31_t)real + (q31_t)imag) >> 1);
330                 imag       = t >> 1;
331             }
332             else {
333                 t          = imag - real;
334                 real       = (real + imag);
335                 imag       = t;
336             }
337             rotation += 1;
338         }
339         /* Calculate tangent value */
340         t = (q15_t)((q31_t)(imag << 15) / real);
341         out = (q15_t)0;
342         for (i = 4; i >=0; i--) {
343             out += coeff[i];
344             arm_mult_q15(&out, &t, &out, 1);
345         }
346         /* Rotate back to original position, in multiples of pi/4 */
347         /* We're using 1.15 representation, scaled by pi, so pi/4 = 0x2000 */
348         out += (rotation << 13);
349         return out;
350     }
351 }
352 void convert_dft_results(int16_t *dft_results, q15_t *dft_results_q15, q31_t *dft_results_q31) {
353  int8_t      i;
354     /*  Check open circuit */
355     for (i = 0; i < (DFT_RESULTS_COUNT / 2); i++) {
356         if ((dft_results[i] < DFT_RESULTS_OPEN_MAX_THR) &&
357             (dft_results[i] > DFT_RESULTS_OPEN_MIN_THR) &&                /* real part */
358             (dft_results[2 * i + 1] < DFT_RESULTS_OPEN_MAX_THR) &&
359             (dft_results[2 * i + 1] > DFT_RESULTS_OPEN_MIN_THR)) {        /* imaginary part */
360 
361             /* Open circuit, force both real and imaginary parts to 0 */
362             dft_results[i]           = 0;
363             dft_results[2 * i + 1]   = 0;
364         }
365     }
366     /*  Convert to 1.15 format */
367     for (i = 0; i < DFT_RESULTS_COUNT; i++) {
368         dft_results_q15[i] = (q15_t)dft_results[i];
369     }
370     /*  Convert to 1.31 format */
371     arm_q15_to_q31(dft_results_q15, dft_results_q31, DFT_RESULTS_COUNT);
372 }
373 /* Calculates calibrated magnitude.                                    */
374 /* The input values are the measured RCAL magnitude (magnitude_rcal)   */
375 /* and the measured magnitude of the unknown impedance (magnitude_z).  */
376 /* Performs the calculation:                                           */
377 /*       magnitude = magnitude_rcal / magnitude_z * RCAL               */
378 /* Output in custom fixed-point format (28.4).                         */
379 fixed32_t calculate_magnitude(q31_t magnitude_rcal, q31_t magnitude_z) {
380     q63_t       magnitude;
381     fixed32_t   out;
382     magnitude = (q63_t)0;
383     if ((q63_t)0 != magnitude_z) {
384         magnitude = (q63_t)magnitude_rcal * (q63_t)RCAL;
385         /* Shift up for additional precision and rounding */
```

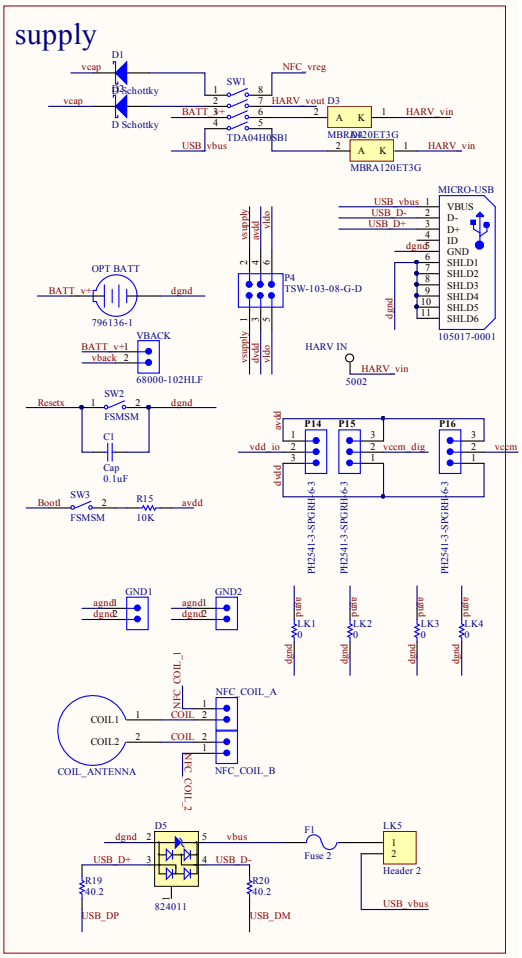
```
386         magnitude = (magnitude << 5) / (q63_t)magnitude_z;
387         /* Rounding */
388         magnitude = (magnitude + 1) >> 1;
389     }
390     /* Saturate if needed */
391     if (magnitude & 0xFFFFFFFF00000000) {
392         /* Cannot be negative */
393         out.full = 0x7FFFFFFF;
394     }
395     else {
396         out.full = magnitude & 0xFFFFFFFF;
397     }
398     return out;
399 }
400 fixed32_t calculate_phase(q15_t phase_rcal, q15_t phase_z) {
401     q63_t       phase;
402     fixed32_t   out;
403 
404     /* Multiply by 180 to convert to degrees */
405     phase = ((q63_t)(phase_z - phase_rcal) * (q63_t)180);
406     /* Round and convert to fixed32_t */
407     out.full = ((phase + (q63_t)0x400) >> 11) & 0xFFFFFFFF;
408 
409     return out;
410 }
411 /* Simple conversion of a fixed32_t variable to string format. */
412 void sprintf_fixed32(char *out, fixed32_t in) {
413     fixed32_t   tmp;
414     if (in.full < 0) {
415         tmp.parts.fpart = (16 - in.parts.fpart) & 0x0F;
416         tmp.parts.ipart = in.parts.ipart;
417         if (0 != in.parts.fpart) {
418             tmp.parts.ipart++;
419         }
420         if (0 == tmp.parts.ipart) {
421             sprintf(out, "      -0.%02d", tmp.parts.fpart * FIXED32_LSB_SIZE);
422         }
423         else {
424             sprintf(out, "%8d.%02d", tmp.parts.ipart, tmp.parts.fpart * FIXED32_LSB_SIZE);
425         }
426     }
427     else {
428         sprintf(out, "%8d.%02d", in.parts.ipart, in.parts.fpart * FIXED32_LSB_SIZE);
429     }
430 }
431 /* Helper function for printing fixed32_t (magnitude & phase) results */
432 void print_MagnitudePhase(char *text, fixed32_t magnitude, fixed32_t phase) {
433     char                tmp[MSG_MAXLEN];
434     sprintf(msg, "    %s = (", text);
435     /* Magnitude */
436     sprintf_fixed32(tmp, magnitude);
437     strcat(msg, tmp);
438     strcat(msg, ", ");
439     /* Phase */
440     sprintf_fixed32(tmp, phase);
441     strcat(msg, tmp);
442     strcat(msg, ")");
443     printString(msg);
444 }
445 void SysTick_Handler(void)
```

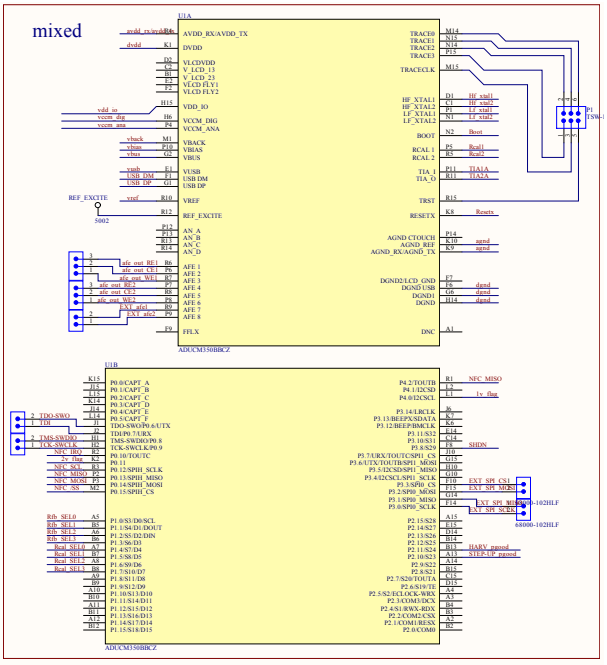
```
{
    /* bump Count up to COUNT  */
    SystickInter = 1;
    SystemExitLowPowerMode(&bCommonInterruptFlag);
}
/*print through UART*/
 void printString(char *ptr){
    sprintf(msg, "%s\r\n", ptr);
    int16_t size = strlen(msg);
    adi_UART_BufTx(hUartDevice, msg, &size);
}
void ConfigureSPI(ADI_SPI_DEV_HANDLE hSpi)
{
  /* data transitions on falling edge of clock */
  if (adi_SPI_SetClockPhase(hSpi, true));
  if (adi_SPI_SetBitrate(hSpi, SPI_CLOCK));
  if (adi_SPI_SetChipSelect(hSpi, SPI_CS_PORT, SPI_CS_PIN));
}
void NFC_SPI_Write_EEPROM(uint8_t address, uint8_t data0, uint8_t data1, uint8_t data2, uint8_t data3)//
     Writes data on NFC Chip EEPROM
{
  ADI_SPI_TRANSCEIVE_TYPE xfr;
  gTransmitBuffer[0] = 0x40;//MODE = 0100000
  gTransmitBuffer[1] = (address<<1);//Word address byte
  /*4bytes data*/
  gTransmitBuffer[2] = data0;
  gTransmitBuffer[3] = data1;
  gTransmitBuffer[4] = data2;
  gTransmitBuffer[5] = data3;
  xfr.pTxData               = gTransmitBuffer;
  xfr.pRxData               = gGarbageBuffer;
  xfr.PrologueSize          = 0;
  xfr.DataSize              = 6;
  xfr.bTxIncrement          = true;
  xfr.bRxIncrement          = false;
  /* submit transaction */
  adi_SPI_MasterTransfer(hSPIDevice, &xfr);
}
```

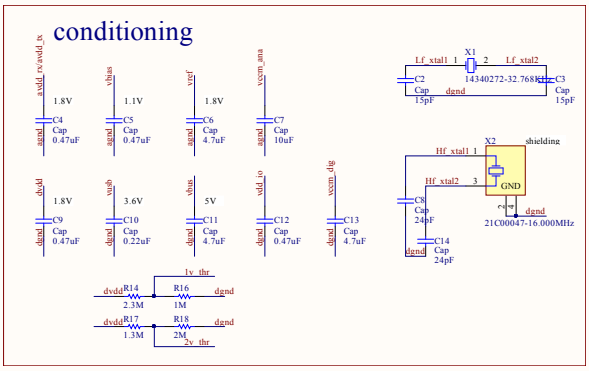
Appendix B

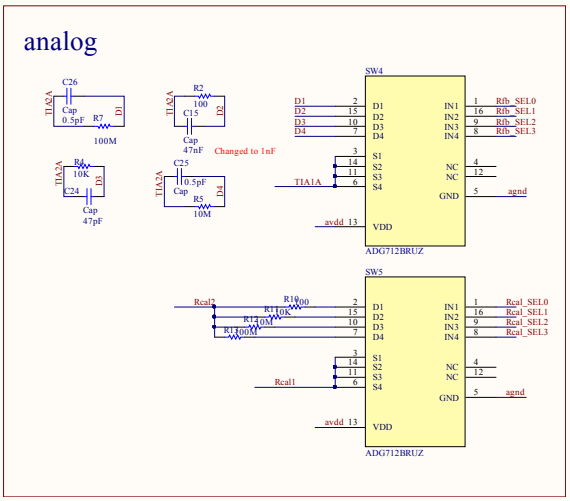
PCB Design

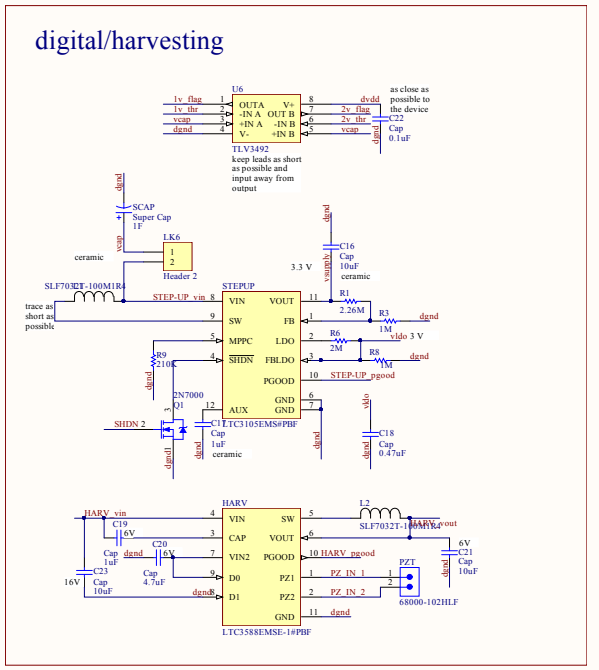
B.1 Schematics

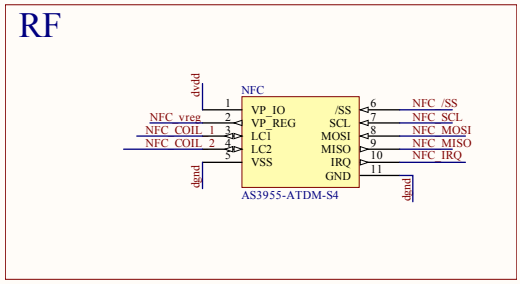




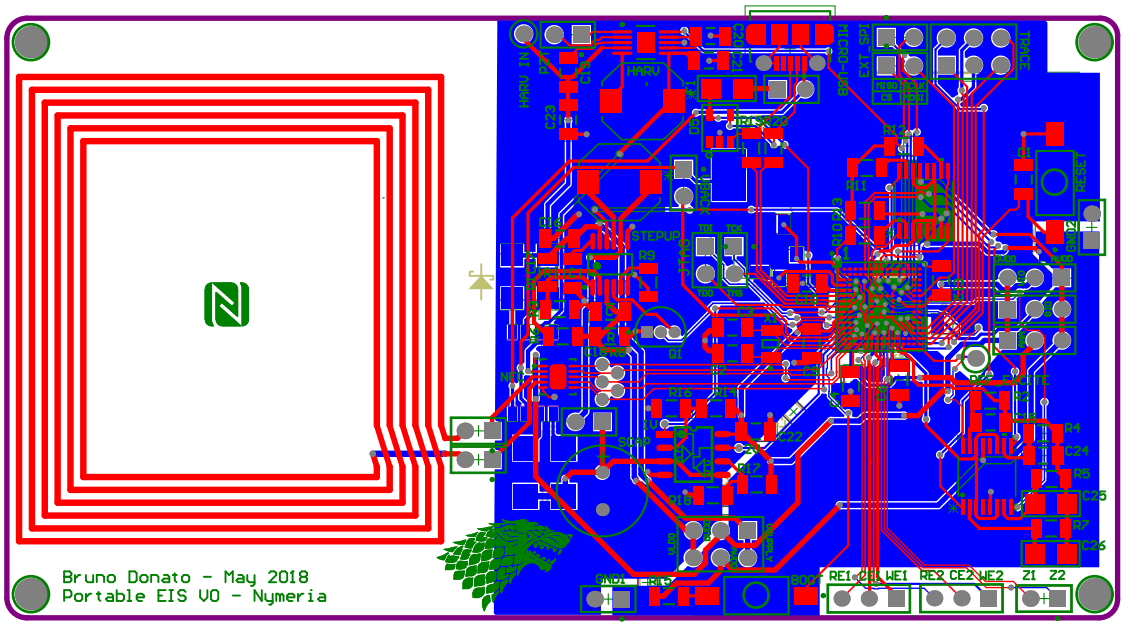

conditioning
avdd_rx/avdd_tx
vbias
vref
vcctn_ana
1.8V
1.1V
1.8V
C4 Cap 0.47uF
C5 Cap 0.47uF
C6 Cap 4.7uF
C7 Cap 10uF
agnd
dvdd
vusb
vbus
vdd_io
vcctn_dig
1.8V
3.6V
5V
C9 Cap 0.47uF
C10 Cap 0.22uF
C11 Cap 4.7uF
C12 Cap 0.47uF
C13 Cap 4.7uF
dgnd
X1
Lf_xtal1
Lf_xtal2
14340272-32.768KHz
C2 Cap 15pF
C3 Cap 15pF
X2
shielding
Hf_xtal1
Hf_xtal2
GND
C8 Cap 24pF
C14 Cap 24pF
21C00047-16.000MHz
1v_thr
R14 2.3M
R16 1M
R17 1.3M
R18 2M
2v_thr

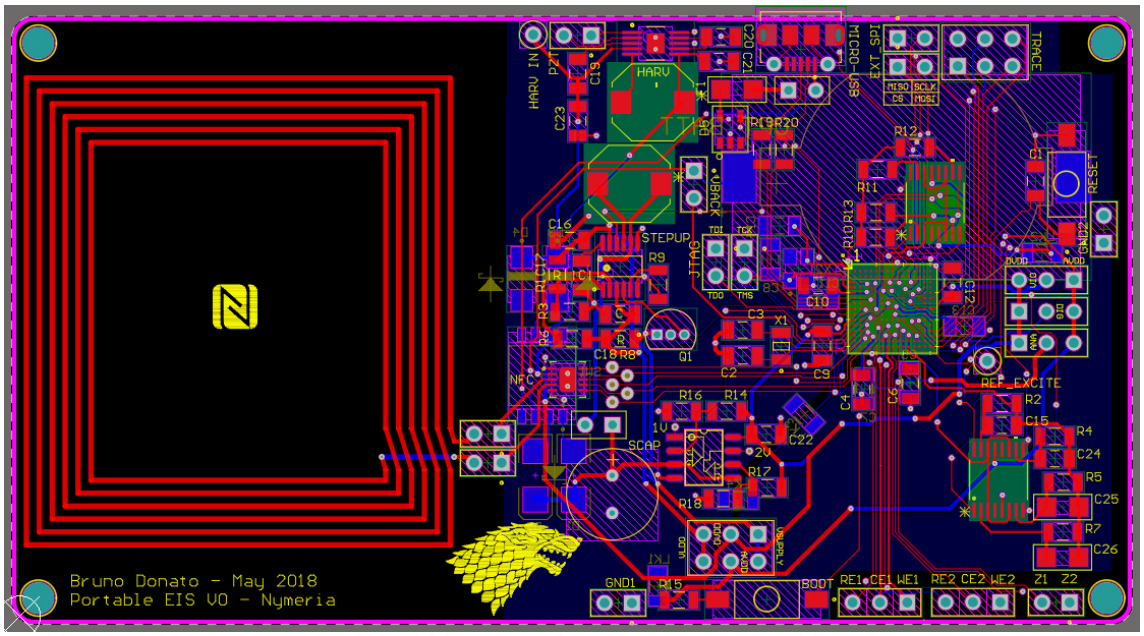

analog
C26 Cap 0.5pF
R7 100M
R2 100
C15 Cap 47nF
Changed to 1nF
R6 10K
C24 Cap 47pF
C25 Cap 0.5pF
R5 10M
TIA1A
TIA2A
D1
D2
D3
D4
SW4
SW5
IN1
IN2
IN3
IN4
S1
S2
S3
S4
NC
GND
VDD
avdd 13
agnd
Rfb_SEL0
Rfb_SEL1
Rfb_SEL2
Rfb_SEL3
Rcal_SEL0
Rcal_SEL1
Rcal_SEL2
Rcal_SEL3
Rcal2
Rcal1
R10 100
R11 10K
R12 1M
R13 10M
ADG712BRUZ


digital/harvesting
U6
TLV3492
keep leads as short as possible and input away from output
as close as possible to the device
1v_flag
1v_thr
vcap
dgnd
OUTA
-IN A
+IN A
V-
V+
OUT B
-IN B
+IN B
2v_flag
2v_thr
dvdd
C22 Cap 0.1uF
SCAP
Super Cap 1F
ceramic
LK6
Header 2
C16 Cap 10uF ceramic
3.3 V
vsupply
SLF7032T-100M1R4
trace as short as possible
STEP-UP_vin
STEPUP
VIN
SW
MPPC
SHDN
PGOOD
AUX
VOUT
FB
LDO
FBLDO
GND
R1 2.26M
R3 1M
R6 2M
R8 1M
vldo 3 V
R9 210k
2N7000
Q1
SHDN 2
LTC3105EMS#PBF
C17 Cap 1uF ceramic
C18 Cap 0.47uF
STEP-UP_pgood
HARV
HARV_vin
HARV_pgood
C19 Cap 1uF 6V
C20 Cap 4.7uF 6V
C23 Cap 10uF 16V
L2
SLF7032T-100MTR4
6V
C21 Cap 10uF
PZT
PZ_IN_1
PZ_IN_2
CAP
VIN2
PGOOD
D0
D1
PZ1
PZ2
68000-102HLF
LTC3588EMSE-1#PBF


RF
NFC
dvdd
VP_IO
VP_REG
LC1
LC2
VSS
/SS
SCL
MOSI
MISO
IRQ
GND
NFC_vreg
NFC_COIL_1
NFC_COIL_2
NFC_/SS
NFC_SCL
NFC_MOSI
NFC_MISO
NFC_IRQ
AS3955-ATDM-S4

B.2 Layout


Bruno Donato - May 2018
Portable EIS V0 - Nymeria


Bruno Donato - May 2018
Portable EIS V0 - Nymeria

Bibliography

[1] J. Jiang, X. Wang, R. Chao, Y. Ren, C. Hu, Z. Xu, and G. L. Liu, "Smartphone based portable bacteria pre-concentrating microfluidic sensor and impedance sensing system," *Sensors and Actuators, B: Chemical*, vol. 193, no. 2014, pp. 653–659, 2014.

[2] H. Jafari, L. Soleymani, and R. Genov, "16-channel CMOS impedance spectroscopy DNA analyzer with dual-slope multiplying ADCs," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 6, no. 5, pp. 468–478, 2012.

[3] D. Sankhala, S. Muthukumar, and S. Prasad, "A Four-Channel Electrical Impedance Spectroscopy Module for Cortisol Biosensing in Sweat-Based Wearable Applications," *SLAS Technology*, 2018.

[4] Y. Yun, Z. Dong, V. N. Shanov, and M. J. Schulz, "Electrochemical impedance measurement of prostate cancer cells using carbon nanotube array electrodes in a microfluidic channel," *Nanotechnology*, vol. 18, no. 46, 2007.

[5] R. Sharma, S. E. Deacon, D. Nowak, S. E. George, M. P. Szymonik, A. A. Tang, D. C. Tomlinson, A. G. Davies, M. J. McPherson, and C. Wälti, "Label-free electrochemical impedance biosensor to detect human interleukin-8 in serum with sub-pg/ml sensitivity," *Biosensors and Bioelectronics*, vol. 80, pp. 607–613, 2016.

[6] S. S. Ghoreishizadeh, X. Zhang, S. Sharma, and P. Georgiou, "Study of Electrochemical Impedance of a Continuous Glucose Monitoring Sensor and its Correlation With Sensor Performance," *IEEE Sensors Letters*, vol. 2, no. 1, pp. 1–4, 2018.

[7] D. I. Stroe, M. Swierczynski, A. I. Stan, V. Knap, R. Teodorescu, and S. J. Andreasen, "Diagnosis of lithium-ion batteries state-of-health based on electrochemical impedance spectroscopy technique," in *Energy Conversion Congress and Exposition (ECCE), 2014 IEEE*, pp. 4576–4582, IEEE, 2014.

[8] R. Masot, M. Alcañiz, A. Fuentes, F. C. Schmidt, J. M. Barat, L. Gil, D. Baigts, R. Martínez-Máñez, and J. Soto, "Design of a low-cost non-destructive system for punctual measurements of salt levels in food products using impedance spectroscopy," *Sensors and Actuators A: Physical*, vol. 158, no. 2, pp. 217–223, 2010.

[9] A. Carullo, F. Ferraris, M. Parvis, A. Vallan, E. Angelini, and P. Spinelli, "Low-cost electrochemical impedance spectroscopy system for corrosion monitoring of metallic antiquities and works of art," *IEEE Transactions on Instrumentation and Measurement*, vol. 49, no. 2, pp. 371–375, 2000.

[10] C. de Beer, P. S. Barendse, and P. Pillay, "Fuel cell condition monitoring using optimized broadband impedance spectroscopy," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 8, pp. 5306–5316, 2015.

[11] M. B. Mejri, H. Baccar, E. Baldrich, F. J. Del Campo, S. Helali, T. Ktari, A. Simonian, M. Aouni, and A. Abdelghani, "Impedance biosensing using phages for bacteria detection: Generation of dual signals as the clue for in-chip assay confirmation," *Biosensors and Bioelectronics*, vol. 26, no. 4, pp. 1261–1267, 2010.

[12] M. Grossi and B. Riccò, "Electrical impedance spectroscopy (eis) for biological analysis and food characterization: a review," *Journal of Sensors and Sensor Systems*, vol. 6, pp. 303–325, 2017.

[13] A. Facchinetti, "Continuous glucose monitoring sensors: past, present and future algorithmic challenges," *Sensors*, vol. 16, no. 12, p. 2093, 2016.

[14] G. De Micheli, S. S. Ghoreishizadeh, C. Boero, F. Valgimigli, and S. Carrara, "An integrated platform for advanced diagnostics," *2011 Design, Automation & Test in Europe*, pp. 1–6, 2011.

[15] B. Donato, F. Stradolini, A. Tuoheti, F. Angiolini, D. Demarchi, G. D. Micheli, and S. Carrara, "Raspberry Pi Driven Flow-Injection System for Electrochemical Continuous Monitoring Platforms," in *IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pp. 288–292, 2017.

[16] F. Stradolini, T. Elboshra, A. Biscontini, G. De Micheli, and S. Carrara, "Simultaneous monitoring of anesthetics and therapeutic compounds with a portable multichannel potentiostat," in *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*, pp. 834–837, IEEE, 2016.

[17] C. Baj-Rossi, T. R. Jost, A. Cavallini, F. Grassi, G. De Micheli, and S. Carrara, "Continuous monitoring of naproxen by a cytochrome p450-based electrochemical sensor," *Biosensors and Bioelectronics*, vol. 53, pp. 283–287, 2014.

[18] R. Wang, Y. Wang, K. Lassiter, Y. Li, B. Hargis, S. Tung, L. Berghman, and W. Bottje, "Interdigitated array microelectrode based impedance immunosensor for detection of avian influenza virus h5n1," *Talanta*, vol. 79, no. 2, pp. 159–164, 2009.

[19] S. Carrara, *Bio/CMOS Interfaces and Co-Design*. Springer Science & Business Media, 2013.

[20] B.-Y. Chang and S.-M. Park, "Electrochemical Impedance Spectroscopy," *Annual Review of Analytical Chemistry*, vol. 3, no. 1, pp. 207–229, 2010.

[21] S. Wetterlin, "A Method of Using Quadrature Sampling to Measure Phase and Magnitude," 2007.

[22] P. M. Ramos, M. Fonseca da Silva, and A. Cruz Serra, "Low-frequency impedance measurement using sine fitting," *Measurement*, vol. 35, no. May, pp. 89–96, 2004.

[23] S. Corbellini and A. Vallan, "Arduino-based portable system for bioelectrical impedance measurement," *IEEE MeMeA 2014 - IEEE International Symposium on Medical Measurements and Applications, Proceedings*, 2014.

[24] S. Grassini, S. Corbellini, E. Angelini, F. Ferraris, and M. Parvis, "Low-cost impedance spectroscopy system based on a logarithmic amplifier," *IEEE Transactions on Instrumentation and Measurement*, vol. 64, no. 5, pp. 1110–1117, 2015.

[25] T. Piasecki, K. Chabowski, and K. Nitsch, "Design, calibration and tests of versatile low frequency impedance analyser based on ARM microcontroller," *Measurement: Journal of the International Measurement Confederation*, vol. 91, pp. 155–161, 2016.

[26] J. Gu, H. Yao, K. Wang, B. Parviz, and B. Otis, "A 10 μ A On-chip Electrochemical Impedance Spectroscopy System for Wearables / Implantables," *IEEE Asian Solid-State Circuit Conference*, vol. 1, pp. 309–312, 2014.

[27] S. Rodriguez, S. Ollmar, M. Waqar, and A. Rusu, "A Batteryless Sensor ASIC for Implantable Bio-Impedance Applications," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 10, no. 3, pp. 533–544, 2016.

[28] J. Hoja and G. Lentka, "Portable Analyzer for Impedance Spectroscopy," *XIX IMEKO World Congress Fundamental and Applied Metrology September 611, 2009, Lisbon, Portugal*, pp. 497–502, 2009.

[29] L. Breniuc, V. David, and C. G. Haba, "Wearable impedance analyzer based on AD5933," *EPE 2014 - Proceedings of the 2014 International Conference and Exposition on Electrical and Power Engineering*, no. Epe, pp. 585–590, 2014.

[30] B. Van Grinsven, T. Vandenryt, S. Duchateau, A. Gaulke, L. Grieten, R. Thoelen, S. Ingebrandt, W. De Ceuninck, and P. Wagner, "Customized impedance spectroscopy device as possible sensor platform for biosensor applications," *Physica Status Solidi (A) Applications and Materials Science*, vol. 207, no. 4, pp. 919–923, 2010.

[31] B. O'Flynn, M. De Donno, C. Barrett, C. Robinson, and A. O. Riordan, "Smart microneedle sensing systems for security in agriculture, food and the environment (SAFE)," *Proceedings of IEEE Sensors*, vol. 2017-Decem, pp. 1–3, 2017.

[32] J. A. Paradiso and T. Starner, "Energy scavenging for mobile and wireless electronics," *IEEE Pervasive Computing*, vol. 4, no. 1, pp. 18–27, 2005.

[33] G. Görge, M. Kirstein, and R. Erbel, "Microgenerators for energy autarkic pacemakers and defibrillators: Fact or fiction?," *Herz*, vol. 26, no. 1, pp. 64–68, 2001.

[34] E. Romero, R. O. Warrington, and M. R. Neuman, "Energy scavenging sources for biomedical sensors," *Physiological Measurement*, vol. 30, no. 9, 2009.

[35] Q. Zheng, B. Shi, Z. Li, and Z. L. Wang, "Recent Progress on Piezoelectric and Triboelectric Energy Harvesters in Biomedical Systems," *Advanced Science*, vol. 4, no. 7, pp. 1–23, 2017.

[36] M. Gorlatova, J. Sarik, G. Grebla, M. Cong, I. Kymissis, and G. Zussman, "Movers and Shakers: Kinetic Energy Harvesting for the Internet of Things," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 8, pp. 1624–1639, 2015.

[37] W.-H. Chen, P.-H. Wu, X.-D. Wang, and Y.-L. Lin, "Power output and efficiency of a thermoelectric generator under temperature control," *Energy Conversion and Management*, vol. 127, pp. 404–415, 2016.

[38] H. Wang, A. Jasim, and X. Chen, "Energy harvesting technologies in roadway and bridge for different applications A comprehensive review," *Applied Energy*, vol. 212, no. December 2017, pp. 1083–1094, 2018.

[39] N. S. Shenck and J. A. Paradiso, "Energy scavenging with shoe-mounted piezoelectrics," *IEEE micro*, vol. 21, no. 3, pp. 30–42, 2001.

[40] A. Almusallam, R. Torah, D. Zhu, M. Tudor, and S. Beeby, "Screen-printed piezoelectric shoe-insole energy harvester using an improved flexible pzt-polymer composites," in *Journal of Physics: Conference Series*, vol. 476, p. 012108, IOP Publishing, 2013.

[41] O. Yaglioglu, *Modeling and design considerations for a micro-hydraulic piezoelectric power generator*. PhD thesis, Massachusetts Institute of Technology, 2002.

[42] W. Wang, T. Yang, X. Chen, and X. Yao, "Vibration energy harvesting using a piezoelectric circular diaphragm array," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 59, no. 9, pp. 2022–2026, 2012.

[43] P. Li, S. Gao, and B. Cong, "Theoretical modeling , simulation and experimental study of hybrid piezoelectric and electromagnetic energy harvester Theoretical modeling , simulation and experimental study of hybrid piezoelectric and electromagnetic energy harvester," vol. 035017, 2018.

[44] R. Hamid and M. R. Yuce, "Sensors and Actuators A : Physical A wearable energy harvester unit using piezoelectric electromagnetic hybrid technique," *Sensors & Actuators: A. Physical*, vol. 257, pp. 198–207, 2017.

[45] G. De Pasquale, A. Somà, and F. Fraccarollo, "Comparison between piezoelectric and magnetic strategies for wearable energy harvesting," *Journal of Physics: Conference Series*, vol. 476, no. 1, 2013.

[46] W. Wang, J. Cao, N. Zhang, J. Lin, and W. H. Liao, "Magnetic-spring based energy harvesting from human motions: Design, modeling and experiments," *Energy Conversion and Management*, vol. 132, pp. 189–197, 2017.

[47] M. Cueff and S. Basrour, "A multi degree of freedom vibration magnetic energy harvester for transport application," *Journal of Physics: Conference Series*, vol. 476, no. 1, 2013.

[48] S. Niu, X. Wang, F. Yi, Y. S. Zhou, and Z. L. Wang, "A universal self-charging system driven by random biomechanical energy for sustainable operation of mobile electronics," *Nature Communications*, vol. 6, pp. 1–8, 2015.

[49] T. Quan, X. Wang, Z. L. Wang, and Y. Yang, "Hybridized Electromagnetic-Triboelectric Nanogenerator for a Self-Powered Electronic Watch," *ACS Nano*, vol. 9, no. 12, pp. 12301–12310, 2015.

[50] S. S. Ghoreishizadeh, I. Taurino, G. De Micheli, S. Carrara, and P. Georgiou, "A Differential Electrochemical Readout ASIC with Heterogeneous Integration of Bio-Nano Sensors for Amperometric Sensing," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 11, no. 5, pp. 1148–1159, 2017.

[51] F. Semiconductor, "FRDM-KL26Z User's Guide," 2013.

[52] D. Ma, C. Mason, and S. S. Ghoreishizadeh, "A wireless system for continuous in-mouth ph monitoring," in *Biomedical Circuits and Systems Conference (BioCAS), 2017 IEEE*, pp. 1–4, IEEE, 2017.

[53] S. Microelectronics, "How to design a 13.56 MHz customized antenna for ST25 NFC / RFID Tags." https://www.st.com/content/ccc/resource/technical/document/application_note/d9/29/ad/cc/04/7c/4c/1e/CD00221490.pdf/files/CD00221490.pdf/jcr:content/translations/en.CD00221490.pdf, 2016. [Online].

[54] Panasonic, "NFC Design Navigator." https://b2bsol.panasonic.biz/semi-spt/apl/en/tool/nfcdesignnavigator/, 2012. [Online].

[55] A. Devices, "Selection table for Energy Harvesting." http://www.analog.com/en/parametricsearch/11503, 2018. [Online].

[56] Eurocircuits, "PCB Calculator." https://be.eurocircuits.com/shop/orders/configurator.aspx, 2018. [Online].

[57] J. Vivari, "3 Steps To Successful Solder Paste Selection," *Proceedings - 2008 International Symposium on Microelectronics, IMAPS 2008*, pp. 611–614, 2008.