A SYSTEM FOR ELECTROCHEMICAL IMPEDANCE SPECTROSCOPY OF ELECTROCHEMICAL SENSORS

VINNOTHINI RASALINGAM

Submitted in the fulfillment of requirements for the degree of
Master of Science
Analogue and Digital Integrated Circuit Design
Of Imperial College London

Department of Electrical and Electronic Engineering
Imperial College London
September 7, 2018

ABSTRACT

This thesis presents the work of a hardware based system designed for complex impedance measurement of electrochemical sensors. The system has a low excitation voltage of 10mVac and with its 10ohms to 100kohms detection range, it is suitable for low impedance analysis. By performing a frequency sweep from 10Hz to 100kHz, the system generates the output in the form of Bode and Nyquist plots of impedance. The complete process of individual circuit block design, component selection, testing procedure, code development and also final measurement results is detailed in this thesis. The final measurement results reveal good accuracy when compared to theoretically calculated values. Built on a 9.85 cm x 5.32 cm sized PCB and consuming 85.4mW of power the system is very much suitable for low power, portable applications.

ACKNOWLEDGMENTS

I wish to this opportunity to thank all those people who have helped me throughout the course of this project. My sincere appreciation to my supervisor Dr. Sara for guiding me well through this project. I am thankful for this opportunity of working with her as I have learnt a lot from her. I am deeply grateful to Vic, Amine and May of Electronics Lab for all the help rendered during my PCB assembly and testing period. Their ever-welcoming presence in the lab and kindness in guiding me made me feel less worried about soldering mistakes and not being able to measure my signals on the scope. I wish to thank the EE Department stores staff for always promptly placing the orders for my components and ensuring I receive them on time. I wish to thank Wiesia who oversees my MSc program for always being approachable and helpful to all of us. I wish to express my sincere gratitude to the government of Malaysia for sponsoring my studies over here, without which none of this would be possible.

Lastly, I want to thank my parents for all the prayers, support and encouragement given to me. My sister, for always believing in me. And my best friend throughout this journey, Divyan, for always being there for me, in all aspects.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF FIGURES

Figure Page

CHAPTER 1

INTRODUCTION

1.1 Project Motivation

The motivation of this project is to develop a system that measures complex impedance for electrochemical cells. Such a system would be extremely useful in the application of electrochemical impedance spectroscopy (EIS). The convenience of having a small portable system would enable such measurements to be carried out almost everywhere instead of the current available set-ups using commercial equipment that requires measurements to only be carried out in the lab environment. A study performed by Zhang et.al.[1] showed that electrochemical impedance of a glucose sensor is related to the sensor's performance over time. The paper indicates that its finding if implemented in a real time platform that performs electrochemical impedance spectroscopy using similar conditions, can pave the way towards sensor self-calibration. The goal of this project is to create a portable system that enables impedance spectroscopy of electrochemical sensors. An application of this project would be to measure the impedance of a sensor and use its data to determine the sensor's state of health. The main objective is to develop a low power, portable system that performs impedance measurements across a wide range of frequency with low excitation voltage (2-10mV).

1.2 Project overview

The diagram below summarizes the steps taken to execute this project. Literature review was carried out to become aware of the state of art designs available in this field of application. A comparison between different system architecture was studied and a system was designed by considering the technical complexity and the effective timeline of execution (step 3 onwards) which is approximately 3 months. Components were sourced to build the hardware system. Drawing of the PCB schematic and layout were carried out using Altium software (refer to appendix for the final layout and schematics). While waiting for the fabrication of the PCB, the chosen hardware's operation guide were studied and test codes were developed. All hardware coding were carried out using Atmel Studio 7. This is followed by PCB assembly and testing of individual component once the PCB was received. Once all individual circuit block testing were completed, the full EIS algorithm was developed and tested. Lastly, basing upon the test results, further improvements were made on the algorithm to improve on the measurement accuracy

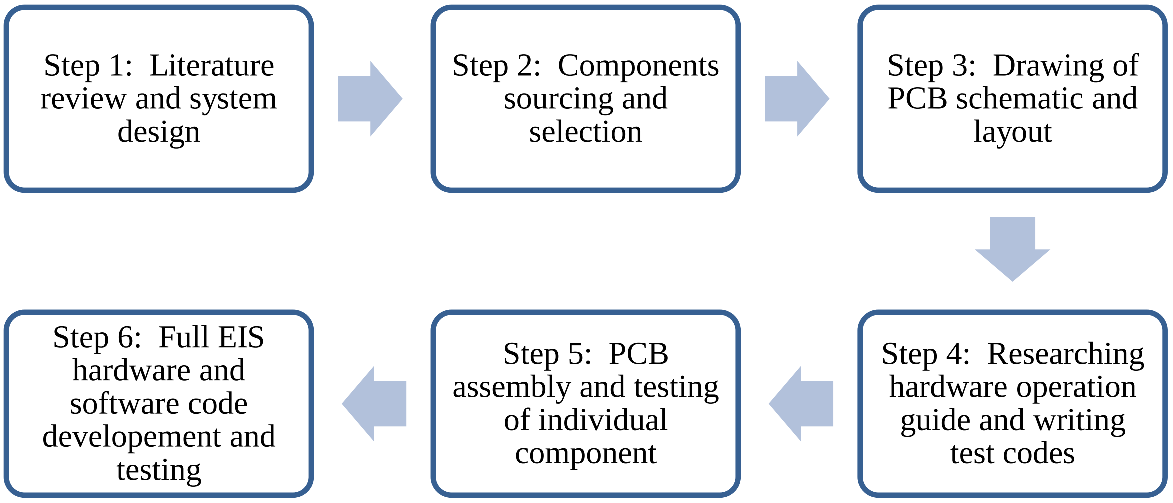

Step 1: Literature review and system design
Step 2: Components sourcing and selection
Step 3: Drawing of PCB schematic and layout
Step 4: Researching hardware operation guide and writing test codes
Step 5: PCB assembly and testing of individual component
Step 6: Full EIS hardware and software code developement and testing

Figure 1: Project execution steps

1.3 Thesis Structure

Chapter 2 provides some background information on electrochemical impedance spectroscopy (EIS) that was needed to design a relevant system that carries out this technique. It provides a basic introduction to the concept of EIS, EIS circuit model, an example of latest EIS application and also EIS data presentation.

Chapter 3 consist of literature review which covers the previous work done in the field of study of this project. The materials reviewed in this section both directly and indirectly contributed to the design process of this project.

Chapter 4 provides an overview of the system as well as the details of the initial design and considerations made.

Chapter 5 describes the PCB implementation of the project. The selection process of individual PCB components is explained in here. This section includes initial testing procedure and results. Issues faced during hardware implementation and the corresponding solutions are also discussed here.

Chapter 6 covers the measurement results of the final designed system. Discussion of accuracy issues and improvements that led to the final design output.

Chapter 7 summarizes the key findings and achievements of the project and concludes the thesis.

CHAPTER 2

BACKGROUND

2.1 Electrochemical impedance Spectroscopy (EIS)

Electrochemical impedance is usually measured by applying an AC potential to an electrochemical cell and then measuring the current through the cell. A sinusoidal potential excitation is applied and measurements are taken in the frequency domain. The response to this potential is an AC current signal. This current signal can be analyzed as a sum of sinusoidal functions (a Fourier series) [2].

EIS is performed by sweeping through a wide range of frequencies (from 10^{-4} to 10^{8} Hz) at a single excitation amplitude. The excitation amplitude is small (2-10mV) and the current response this will be a sinusoid at the same frequency but shifted in phase for a complex impedance. Figure 2 below shows an example of input excitation signal and the response current of a capacitor. It can be observed here that the current leads the input potential.

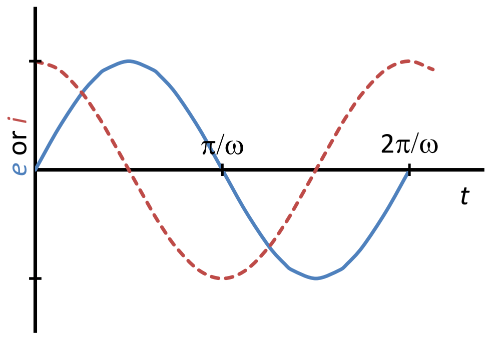


Figure 2 : Example of an input excitation signal, e and resulting current response, I of a capacitor.

Impedance is defined as the opposition of an electrical system to the flow of electric current and carries units of Ohms, Ω. It is known as resistance if there is no phase shift in current under an applied potential, and vice versa and if its impedance is not a function of frequency. Under these conditions, the Ohm's Law applies:

$$V{=}IR \ or \ R{=}VI$$

If we have an excitation signal with the following function in time:

$$E_t = E_o$$

The current response, I_t, would have a phase shift (Φ) and a different amplitude than I_0.

$$I_t = I_o$$

The resulting impedance of the system with magnitude, Z_o, and a phase shift, Φ is expressed as the following:

$$Z(\omega) = \frac{Et}{I_t} = \frac{Eosin\ (\omega t)}{Io\sin(\omega t + \emptyset)} = Zo\frac{sin\ (\omega t)}{\sin(\omega t + \emptyset)}$$

2.2 EIS Data presentation and Electrochemical Cell Models

EIS is commonly represented using Nyquist and Bode plots. The expression for complex impedance Z(ω) is composed both real and an imaginary part. Nyquist Plot (as shown in Figure 2) refers to the real part is plotted on the x-axis and the imaginary part is plotted on the y-axis of a chart. The y-axis of this plot is usually the negative of imaginary part and that each point on the Nyquist Plot is the impedance at one frequency. Frequency information is not obvious from Nyquist plot.

Bode Plot refers to the impedance being plotted with frequency in logarithmic scale on the x-axis and both absolute values of the impedance ($|Z|=Z_0$) and the phase-shift on the y-axis. Bode Plot clearly shows frequency information.

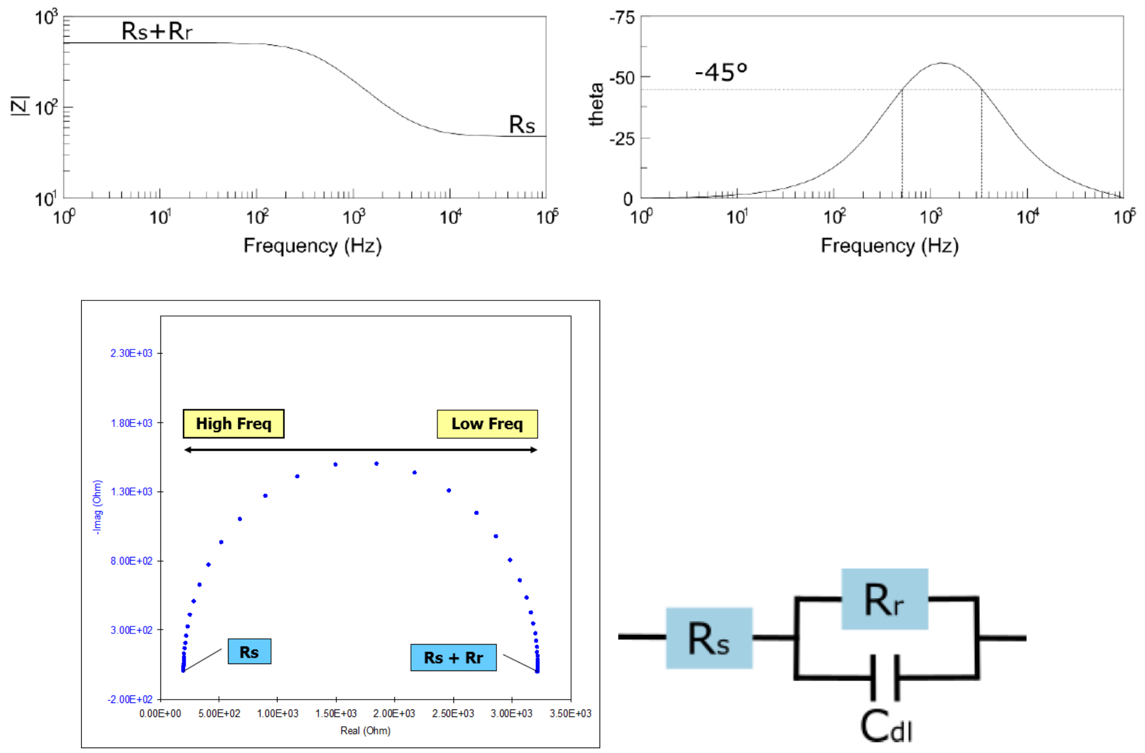


Figure 3: Top -Bode plot of simplified Randles cell; Bottom left- Nyquist plot of simplified Randles cell shown on bottom right.

The Simplified Randles cell (refer figure 3) is one of most common cell models. It represents the solution resistance, R_s, double layer capacitance C_{dl} and a charge transfer (or polarization resistance), R_{ct}. The double-layer capacitance is in parallel with the charge-transfer resistance.

These are also the parameters that can be extracted from an EIS measurement. R_{ct} is only part of a combined resistance lumped together as Rr in the Simplified Randles Cell. The solution resistance refers to the electrolyte resistance between the electrodes namely the reference electrode and the working electrodes. The resistance of an ionic solution depends on the ionic concentration, type of ions, temperature, and the geometry of the area in which current is carried. The electrical double layer on the interface between a charged electrode and its surrounding electrolyte forms the double layer capacitance, C_{dl}. The charge transfer reaction in the electrochemical cell has a certain speed. The speed depends on the kind of reaction, the temperature, the concentration of the reaction products and the biasing potential. This reaction forms the charge transfer resistance, R_{ct}.

The equivalent impedance equation of a Simplified Randles Cell is as following [3]:

$$Zeq = Rs + \frac{1}{\frac{1}{Rr} + j\omega Cdl} = Rs + \frac{Rr}{1 + j\omega RrCdl} \qquad \text{(Equation 1)}$$

And this can be resolved to real and imaginary portions as shown below.

$$Z_{RE} = Rs + \frac{Rr}{1 + \omega^2 Rr^2 C_{dl^2}} \qquad Z_{lm} = \frac{-j\omega C_{dl^2} Rr^2}{1 + j\omega^2 Rr^2 C_{dl^2}} \qquad \text{(Equation 2)}$$

Negative of Z_{Im} forms the y-axis of the Nyquist plot while Z_{Re} forms the x-axis.

To obtain the magnitude and phase portions of the Simplified Randles Circuit, the same equations 1 and 2 can be used in the following way:

$$\| Z \| = \sqrt{Z_{Re^2} + Z_{lm^2}}$$

$$\varphi = \tan^{-1}\left(\frac{Z_{lm}}{Z_{Re}}\right)$$

On a Bode plot, both |Z|, the magnitude and phase forms the x-axis while the frequency in logarithmic scale forms the y-axis. These were the equations that were used to calculate the theoretical values of the R-C network used in the testing of the final system in this project.

The parameters in a Randles circuit can be extracted from the Nyquist plot. Refering to Fig. 3, the high-frequency intercept yields Rs, which forms the frequency-independent portion of the model. The low-frequency intercept gives Rs+Rr, and, therefore Rr, after subtracting Rs. This is the characteristic impedance of the feature. The characteristic capacitance of this feature is found using Rr and ωc with the following formula:

$$C=1/(Rr*\omega c).$$

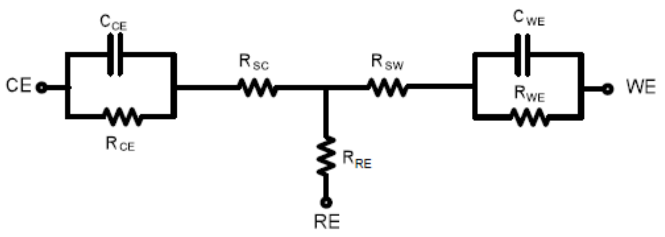


Figure 4 :RC-equivalent model of a three-electrode based electrochemical sensor used for circuit simulations [4].

The RC model in Figure 4 is one of a three-electrode based electrochemical sensor used to simulate the system behaviour and the stability of the a sensor readout circuit when it interfaces with a sensor, the small-signal circuit model of the electrode–electrolyte interfaces in an electrochemical cell [4]. In this project however, Simplified Randles cell based RC network was used instead of the circuit in Figure 4 for testing purposes.

2.3 Application of EIS

One of most recent applications of EIS would be cortisol bio-sensing in human sweat based wearable application [5]. The author presents detection of cortisol in human sweat and studying the effects of pH and temperature variations on it through EIS measurements. The paper also demonstrates impedance measurements on four sensors at 1 kHz through time division multiplexed (TDM) using state-of-the art Analog Device's ADuCM350 Low Power Meter On A Chip. More examples of EIS applications will be discussed in the next chapter.

In conclusion, EIS is a versatile technique of obtaining information on the state of electrochemical cell, it is non-destructive and has high information content.

CHAPTER 3

LITERATURE REVIEW

This chapter details some of the previous works done by various authors that are related to the topics explored in this project. It is divided into three sections namely papers on electrochemical sensor readout circuit design, papers on EIS systems designed using hardware-software based systems and papers on EIS systems designed on integrated chip (IC). Along with the implementation, the application of EIS is also discussed.

3.1 Paper on readout circuit for sensor

Paper by S. Ghoreishizadeh et al. provided good insight on biosensor readout circuits specifically chronoamperometry (CA) and cyclic voltammetry (CV) readout circuits and how the designed circuits have been tested. Some understanding of how the three electrodes of the sensor could be biased was also obtained from this paper. The three electrodes are called working electrode (WE), reference electrode (RE), and counter electrode (CE). According to the paper, the electrochemical reaction happens at the WE, the RE functions to control the voltage of the solution while no current flow is allowed on this electrode, and the CE is to close the loop for the sensor current. CA and CV measure a current out of a terminal (WE or CE) when a fixed or variable voltage is applied to the other two electrodes, respectively.

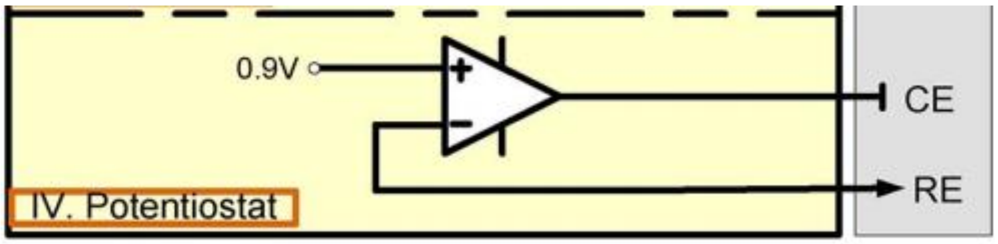


Figure 5: Potentiostat configuration from the paper [6]

A potentiostat circuit keeps the electric voltage of the WE to a value that can be fixed or variable with respect to RE using a negative feedback around RE. The current that flows between WE and CE is collected using a transimpedance amplifier connected after the potentiostat stage [7].

On the part of testing the readout circuit, this paper detailed the testing of its CV readout. To test the circuit, an internally generated triangular waveform is applied to the positive terminal of the op-amp, while the negative terminal of the op-amp with WE terminal is connected to a resistor. The other terminal of it is connected to CE terminal of the potentiostat, while the potentiostat is connected as a unity gain buffer (i.e., RE and CE are short connected) and a voltage of 0.9 V is applied to its positive terminal. Therefore, the input current of the CV readout circuit can be calculated by the formula below:

$$\mathrm{Ires} = \frac{Vramp - 0.9}{Rtest}$$

A similar test setup in connecting the potentiostat as this was adapted in this project.

3.2 Paper on Implementation of EIS on IC

Jingren Gu et al. presents a fully integrated on chip EIS system [7]. This paper saves power and area by omitting the need for analog filters in their novel method of using time-domain integration method to realize Electrochemical Impedance Spectroscopy (EIS).

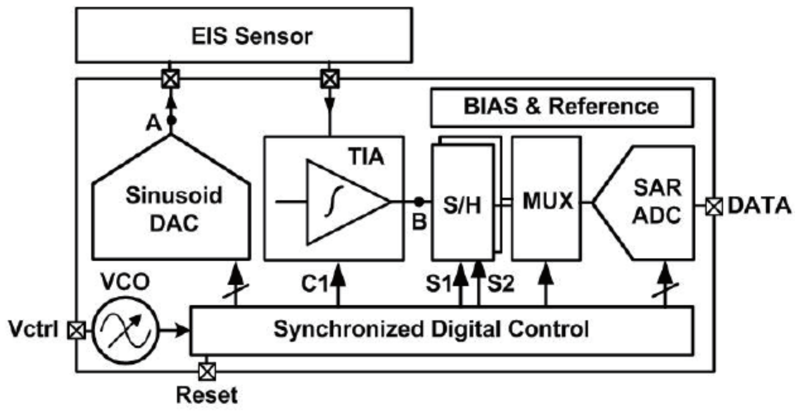
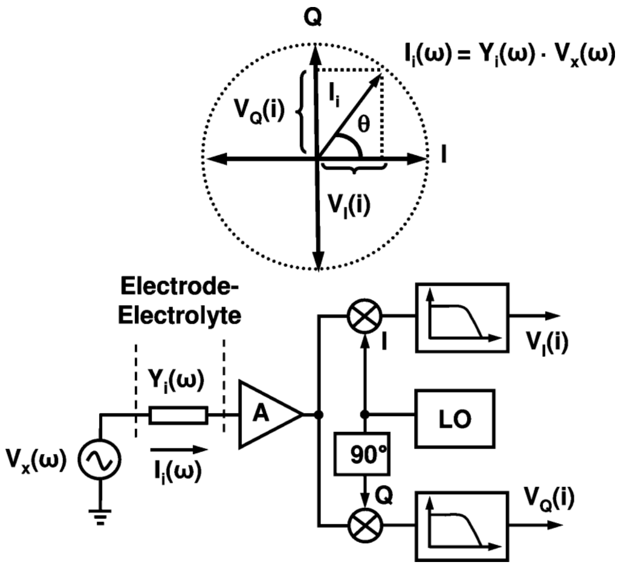


Figure 6: Block diagram of EIS system using time-domain integration method. Printed from [7]

The EIS system in this chip consists of sinusoid DAC but without a low pass filter at the output, allowing high frequency harmonics to remain at the output. The reason for this is that the time-domain integration detection method employed which can attenuate the error caused by those harmonics. The response current is sensed by a switched capacitor integrator-based trans-impedance amplifier (TIA) with control synchronized with the sinusoid DAC. This is followed by a sample and hold circuit and the integration output is sampled and digitized by an 8-bit SAR ADC. This system performs frequency sweep from 900Hz to 2.8 kHz and consumes 10uA under a 1.2V supply. After testing, the system reports an error of below 5% for amplitude and below 8% for phase. Another popular work is that of Manickam et al. which presents a fully integrated, 4-metal layered, electrochemical impedance spectroscopy (EIS) biosensor array on 0.35um CMOS technology[8]. The chip works by measuring in real time the admittance of electrode-electrolyte interface, in each biosensor pixel over a wide range of frequencies (10Hz–50MHz) with adjustable amplitude from 1 to 100mV. It establishes a controlled sinusoidal excitation signal across the electrode interface and measures both the amplitude and the phase of the generated current, in individual pixels. Subsequently computing each pixel's interface admittance using both the generated current and the excitation voltage. Phase and amplitude are measured using a low-noise transimpedance amplifier (TIA) and two quadrature phase mixers. The orthogonal I and Q signals are created by an on-chip digital quadrature generator using an external LO. The I signal is phase-locked to the excitation voltage. Each pixel has two DC outputs, V_I and V_Q which also contribute to the calculation of the admittance. Finally the following figure shows an illustration of the impedance detection methodology based on finding the admittance Y(w) and the formula is used to calculate the admittance:



$$|Y_i(\omega)| = \frac{\sqrt{V_I^2(i) + V_Q^2(i)}}{A \cdot |V_x(\omega)|}$$

and

$$\angle Y_i(\omega) = \tan^{-1}\left(\frac{V_Q(i)}{V_I(i)}\right),$$

Figure 7 : Example of impedance detection method as proposed in [8].

3.3 Papers on implementation of EIS using hardware-software integration

This section is on previous works that designed and implemented EIS systems on PCB. Lang Yang et al. reports a circuit design on PCB for signal stimulus generation and response signal acquisition for EIS [9]. The FFT signal processing however is done off board in this design.

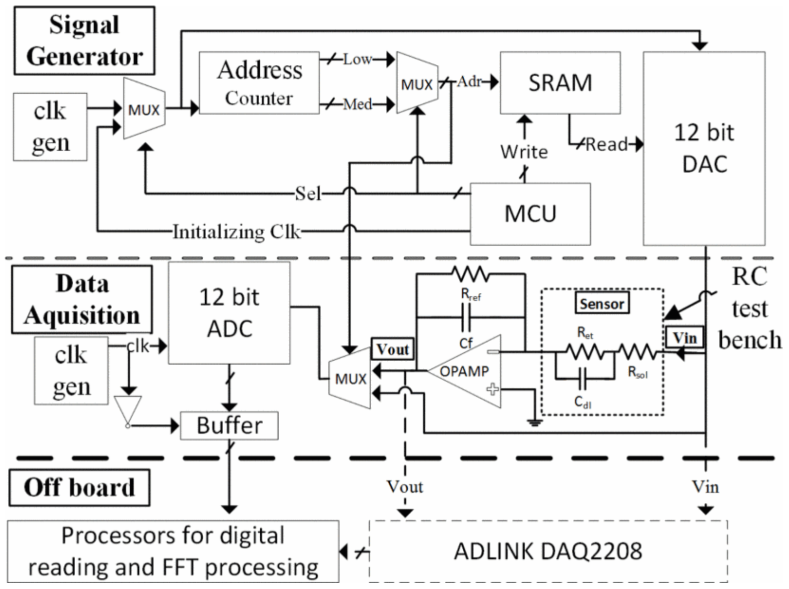


Figure 8: EIS Hardware implementation where the FFT processing is done off-board. Printed from [9].

The circuit has achieved the specs of 9.2 Ohms limit of detection with frequency range of the stimulus signal ranging from 2Hz to 2kHz of frequency and approximately 10.02mV of amplitude. The hardware in this design consists of stimulus signal is generated using a clock, address counter, SRAM and 12-bit DAC with all of them controlled by a MCU. The waveform output from the DAC is applied to the sensor. On the signal acquisition part, a TIA is used to convert the response current to an output voltage, it is followed by a 12-bit ADC which digitizes the sensor response signal before it is sent to a computer for processing. Their method of measuring the sensor impedance was to sample the Vin and Vout separately and convert them to digital outputs. They then performed fast Fourier transform (FFT) on the digital outputs before calculating the impedance's magnitude and phase offline.

In summary, this paper gives a good overview of the fundamental components of an EIS circuit and a method of calculating complex impedance using both the input and output signal information.

J. Punter-Villagrasa et al. presented an EIS system for blood analysis that detects impedance in the working range of 10Hz to 100kHz [10]. The system is targeted to be used with three electrode sensors.

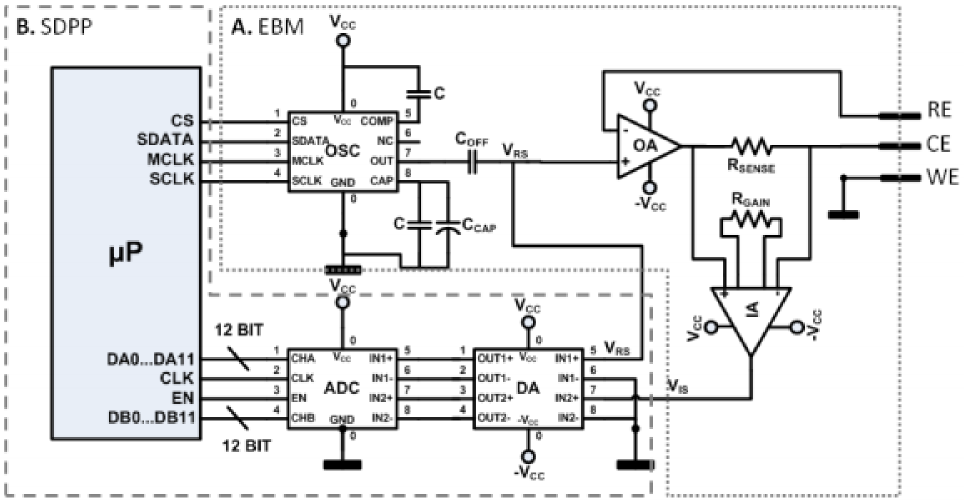


Figure 9: EIS hardware implementation for blood analysis using embedded lock-in software. Printed from[10].

Their impedance analysing circuitry consists of three main portions, namely the signal generation and electrodes biasing portion, signal digitizing and processing portion and finally Digital Lock-in method was used to provide with the impedance results through Nyquist and Bode plots. The signal generation part of their circuit was built using a programmable waveform generator AD9833 and the sensor readout circuitry was based on a potentiostat with an instrumentation amplifier current readout. The current through the electrodes is proportional to the current that flows through the R_{sense}. The amplifiers which bias the electrode and the instrumentation amplifier that senses the electrode current were chosen on the basis of having low leakage current, low distortion capable of high output driving to transfer the signal on to the electrode for the former and also having low noise, ultralow bias current for electrode current sensing in the case of latter. The signal digitizing and processing portion consist of an dual, low power ADC which able to convert signals from both analog input (waveform generator output signal and sensor current converted to voltage signal) simultaneously at 40MSPS.

For the impedance calculation, an embedded lock-in software running on a real time platform was used to provide the magnitude and phase results. The system was tested using three different pairs of resistors and capacitors in parallel configuration and the results were plotted against theoretical values. The paper reports an error of 3.5% for the system designed. Subsequently the system was tested using commercial sensors and compared against commercial equipment SP-150 results. As a summary, this paper presents a rather traditional method of impedance measurement by using digital lock-in and with a FRA (Frequency Response Analyzer) approach, however the drawback of this system is its high power consumption at $24V_{DC}$ and 0.35A of current.

This section explores the work of people who have used impedance analyser chip AD5933 to perform EIS. Diming Zhang et al. has developed an AD5933 based hand-held device with smartphone platform to perform EIS for protein detection purposes [11]. The hardware system consist of a Arduino board, amplifier, an external clock that scales the input clock signal to the AD5933 impedance analyser, 9V battery and a Bluetooth module. On the software end, an Android app was developed to display the Nyquist plot of the measurement results.

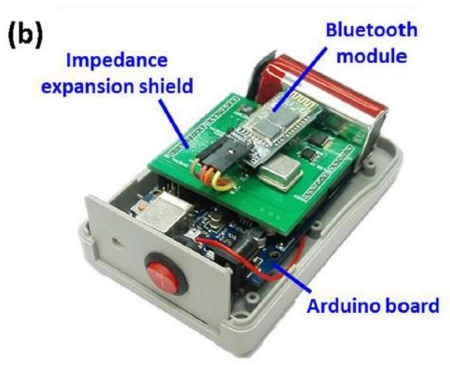


Figure 10: An Arduino-Android platform based EIS system by [11]

The range of frequency sweep for this system is 10Hz to 10kHz, with no mention on the power consumption and excitation voltage amplitude of the system. The system reports to be able to accurately recorded impedance measurement from 10 Hz to 10 kHz, although its Nyquist semicircle in EIS measurement was smaller than that of commercial device such as electrochemical workstation. The reasons given for this were the differences by variations in testing strips and electrochemical fluctuations, and not by difference in performance of the electronic system. Other works that have used AD5933 impedance analyser as the center of their impedance measurement system includes Bog´onez-Franco et.al in biological tissue based impedance measurements. A modified Howland current source was used to inject a current in the range of 10 μAp for frequencies of 100 Hz to 200 kHz and the resultant voltage from the impedance of the tissue is measured using a feedback resistor before being fed into the impedance analyser [12]. The authors found that the measurement error was substantial at low frequencies because of the high impedance of the electrodes used and because down-scaled clock input the impedance analyser chip was not provided.

3.4 Conclusion

The EIS systems reviewed earlier is summarized in the table below:

Implementation	Operating Frequency	Low Power Application	EIS field of application	Impedance analysis method
IC	900Hz-2.8 kHz	Yes 1.2V , 10µA	Not stated	Time-domain integration detection
IC	DC-50MHz	Yes 84.85mW	Biosensors	Measures admittance through lock-in method
Hardware-Software	10Hz - 100kHz	No $24V_{DC}$, 0.35A	Blood analysis	Digital Lock-in Frequency Response Analyzer (FRA)
Hardware-Software	2Hz to 2kHz	Not stated	General	Off board FFT processing
Hardware-Software	10Hz to 10kHz	No 9V	Protein detection	AD5933 impedance analyzer
Hardware-Software	100 Hz to 200 kHz	Yes 2.8V-3.3V	Biological Tissue	AD5933 impedance analyzer

Table 1: Summary of EIS systems in the literature reviewed.

CHAPTER 4

SYSTEM DESIGN

4.1 System Design

Excitation Frequency:	10Hz to 100kHz
Excitation voltage amplitude:	5mV to 10mV
Measurable Impedance:	10Ω to 100kΩ
Power consumption:	Low power (milliwatt range)

Table 2: Specifications of the project.

Table above shows the specifications set for this project. The small excitation voltage amplitude is so that the physical properties of the sensor is not is not destroyed in EIS the process. The low power requirement is so that the system can be portable and used anywhere with USB power source instead of needing higher voltage power supplies from the lab.

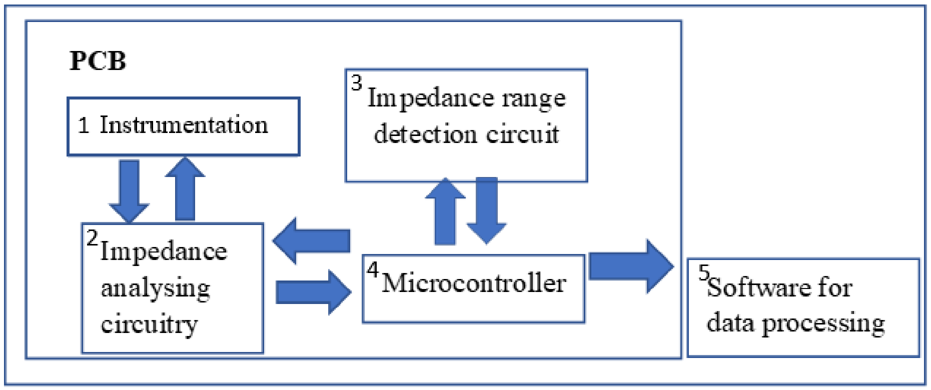

PCB
1 Instrumentation
3 Impedance range detection circuit
2 Impedance analysing circuitry
4 Microcontroller
5 Software for data processing

Figure 11: System Overview

The EIS system consist of a PCB which houses the hardware necessary to interface with the electrochemical sensor as well as the circuit that does the impedance measurement. The system has five parts (blocks) namely analogue front end, impedance analysing circuitry, impedance range controlling circuit and the microcontroller.

These blocks will be described further in this section according to the order of the number labelled in Figure 11.

4.2 Instrumentation

The instrumentation circuitry consists of three op-amps that play the roles of control potentiostat, inverting summing amplifier and TIA (transimpdance amplifier) (refer figure 12). The potentiostat has direct access to the sensor as it biases the RE electrode of the sensor with an AC excitation voltage at a certain DC value. The output of this op-amp is connected to another terminal of the sensor namely the CE electrode. The function of the potentiostat is mainly to hold the two electrodes of the sensors at a certain voltage potential. The third terminal of the electrode, WE is connected to the negative terminal of the TIA. The TIA collects the current that flows from the CE to the WE of the sensor and converts to voltage. The voltage values converted to depends on the feedback resistor value that it is connected to. This voltage is then fed to the impedance analyzing IC.

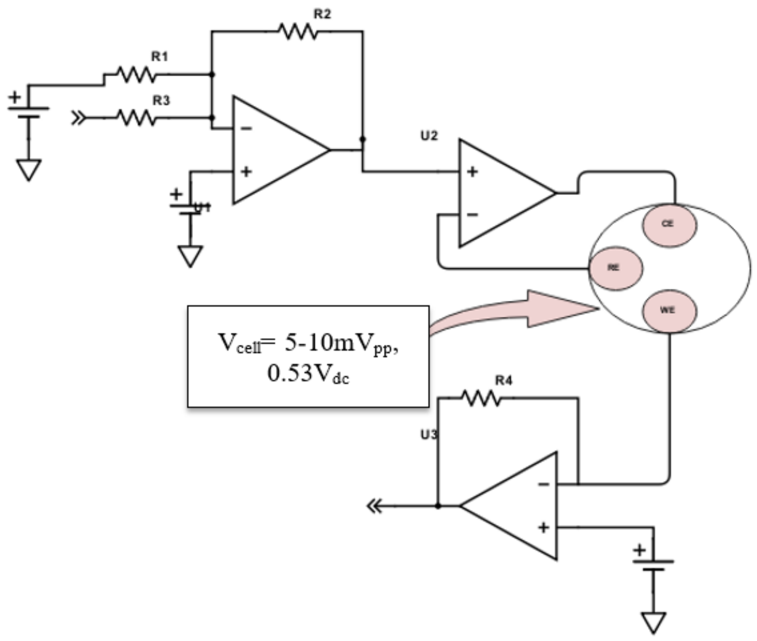


Figure 12 : Initial Instrumentation topology (Topology 1)

Inverting summing amplifier here has two main functions. It performs re-biasing of the DC value and it also attenuates peak to peak value of the AC waveform to provide a suitable excitation voltage to the electrochemical cell. The re-biasing of DC value is needed because the waveform generated by the impedance analyser chip has an offset value of 200mV which does not match the DC biasing of internal op-amp at the receiving end of the impedance analyser chip which is biased at $V_{DD}/2$ (1.65V). This causes a large DC offset difference between the two ends of the unknown impedance. Another reason for re-biasing the circuitry is to ensure that excitation voltage of the sensor has a DC bias of 0.53V [1].

During PCB implementation, while this topology was used for the first 200mV excitation voltage board testing, the inverting summing amplifier circuit had to be replaced by a new amplifier circuit shown below for 10mV board implementation.

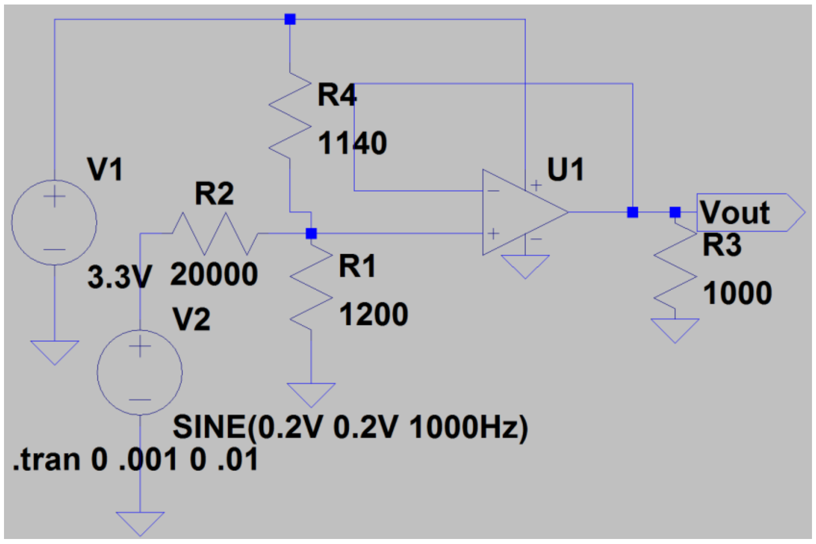


Figure 13: Improved Instrumentation topology SPICE simulation (Topology 2)

The reason for it was that the amplifier did not work well in less than unity gain (gain of <1). Many sources claim that in practice, some amplifiers can become unstable when operating in gain of less than 1. Hence to avoid the risk of oscillations at any point of operation, the circuit was replaced with a new one.

It also becomes not possible to select resistor pair values that satisfies both conditions of attenuating the AC output voltage of AD5933 by a certain value at the same time re-biases the DC offset of the AD5933 output waveform by a different magnitude. Usage of DC blocking capacitor at the output of the AD5933 VOUT pin as advised in the application note [13] is not viable for this project as the operating frequency of this project is very low at 10Hz hence the capacitor size becomes very large and causes distortion of the output waveform.

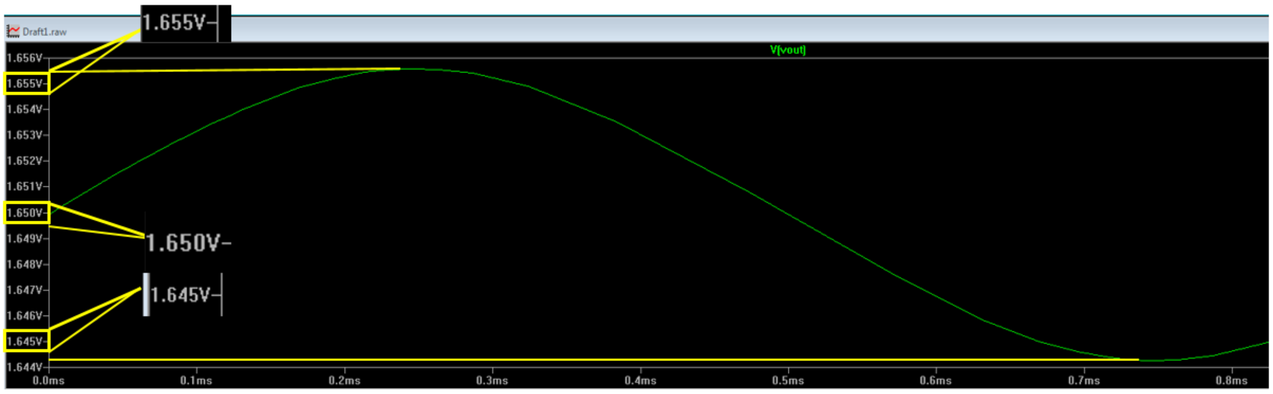


Figure 14: SPICE simulation results of AFE topology 2 (successful in producing 10mVpp at 1.65Vdc)

The new circuit is relatively simpler as it consists of a voltage divider R1, R2 with values chosen such that the waveform generator's input is attenuated from 200mV to 10mV. R4 resistor plays the role of re-biasing the 200mV DC offset from waveform generator's input to 1.65V using the 3.3V signal available on the PCB. R4 value was obtained through SPICE simulation after initially setting R1 and R2 values. Simulation results (Figure 14) shows signal at Vout is a 10mVpp signal biased at 1.65Vdc. Bench results from the PCB of this circuit is shown in Chapter 6.

4.3 Impedance analyzing circuitry

Impedance analyzing circuitry consists of an impedance analyzer integrated chip and external clock, feedback resistor for internal TIA. Due to the short time frame of effective project time, an IC that incorporated a waveform generator, ADC and DFT engine on chip was chosen as it enabled fast prototyping and concept verification versus building the mentioned blocks from scratch.

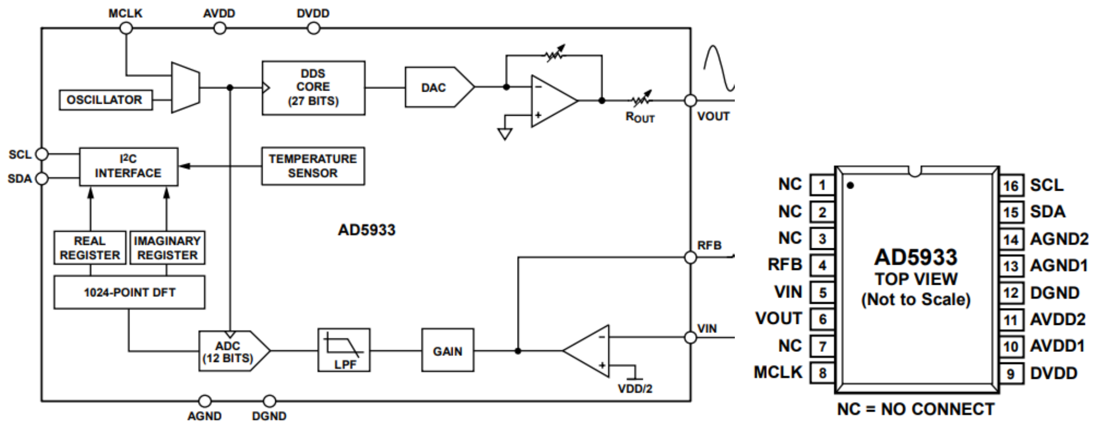


Figure 15: Impedance Analyzer (AD5933) block diagram (left), pinout (right) [14]

An IC that had all the mentioned circuit blocks on-chip is the AD5933 impedance analyzer by Analog Devices. However, its specification does not match the specification of this project (refer to table 3 below). Hence it needs to be interfaced with external components to cater to the needs of this project. Briefly speaking, AD5933 operates by generating sinusoidal waveform that excites an external impedance, it then samples voltages that corresponds to the current generated by an unknown impedance. DFT is performed on the sampled sinusoid signal to give real and imaginary values. A known impedance is used to calibrate the system. The real and imaginary values returned by the chip due to the known impedance becomes the scaling factor to calculate an unknown impedance. The AD5933 comes with an internal reference oscillator of 16.776 MHz which is the clock input to the DDS waveform generator and ADC.

Frequency range	1kHz to 100kHz
Impedance measurement range	1 kΩ to 10 MΩ
Power supply	2.7 V to 5.5 V
AC Output Excitation Voltage with DC bias	2V-200mVpp 2V-200mVdc
DC Output Impedance (depends on voltage excitation range)	200Ω to 2.4kΩ
Internal Oscillator Frequency	16.776 MHz
Power Consumption (in terms of I_{DD} values at 3.3V) Normal mode Standby mode Power down mode	 10mA 11mA 0.7µA

Table 3: Impedance analyzer (AD5933) Main Specifications [14]

The impedance analyzer chip generates sinusoidal waveform using the on chip 27-bits resolution DDS (Direct Digital Synthesizer) DAC and a programmable amplifier circuitry before being sent to the VOUT pin. The 27-bits resolution DDS allows it to be able to perform sweeps with an increment of as low as 0.1Hz. The internal oscillator supports impedance excitation and analysis at frequencies from 1kHz to 100kHz. Based on application note of AD5933 [13], without a scaled clock input, the impedance analyser is only able to accurately analyse frequencies between 10kHz to 100kHz (more on this would in discussed in Chapter 6). Since the specification set for this project is to perform frequency sweep from 10Hz to 100kHz and analyze impedance at this range, clock scaling of the input clock to the DDS is needed to extend the range [15]. An external clock with variable frequency is needed to perform this.

4.5 Impedance range detection circuit

In the literature review section, many papers using AD5933 as part of their impedance analysing circuitry have found promising results, but they did not address the issue of impedance range estimation. In order to use the AD5933 to analyse an unknown impedance accurately, we need to have some idea of what the range of impedance measured is like, only then we can assign a suitable feedback resistor that amplifies the current from the impedance and converts it to voltage. The converted voltage should be in the order of 0 to VDD. Failing to follow this requirement causes the ADC to saturate and not give the right results. Also since method of impedance measurement using AD5933 involves the use of a calibrated known impedance, this presents another reason for us to have some idea of the magnitude of the impedance to be measured.

In this case knowing the magnitude of the impedance helps us assign a suitable calibration resistor for the unknown impedance calculation. For example, if we know that the unknown impedance is in the range of 10kohms to 100kohms, a suitable calibration resistor based on AD5933 application note will be as following [16]:

$R_{cal} = (R_{high}+R_{low})/3$;

When it comes to devising a method to get some idea of the range of impedance measured, the straightforward way would be to sample the voltage output at the I-V amplifier and find out if its within 0 to VDD thereby indicating that the impedance to be measured would not saturate the ADC. The feedback resistor value used at this point will give us an indication of which calibration resistor to use to calculate the final impedance value.

However the voltage at the output of the I-V amplifier is a sinusoidal signal with a frequency ranging from 10Hz to 100kHz, hence by considering the Nyquist sampling theorem, an ADC with a minimum sampling rate of 200kHz is needed. In practice, since we are interested in the peak value of the signal, an accurate ADC with a sampling rate of minimum 200kHz along with some processing would also be needed to ensure we sample the peaks of the sine wave. A much simpler solution would be to use a peak detector circuit that converts the sinewaves to a flat signal representing the peak value of the circuit.

Peak detector based solution

The impedance range detection circuit consist of a MUX, peak detector circuit and the microcontroller's ADC. It forms a feedback loop to tell the system which range of impedance it is encountering. Peak detector circuit is used for impedance magnitude detection. Peak detector circuit measures the peak voltage just before it enters the impedance analyzer chip which is also the point after the feedback resistor. It checks if voltage is between 0.3V to 3.0V. If the voltage isn't in this range, the program would assign the next biggest value resistor to the output of the I-V convertor amplifier and this goes on until the right resistor is assigned to the output of the amplifier. This is to ensure that the right calibration resistor's gain factor is used eventually in the software processing to calculate the unknown impedance's value.

The ADC would sample the voltage output at the peak detector circuit, the microcontroller would compare the sampled values to the pre-programmed ADC limits which forms digital value that corresponds to the near 0 and VDD values. This is initially carried out with the lowest value feedback resistor so that if the impedance is around the range of 10 to 100ohm, the high current generated will not damage the circuit. If the sampled value does not fit this range, the microcontroller sends signal to the MUX to assign the next highest value resistor to the output. This goes on until the right feedback resistor of the four available resistors is assigned to the output.

A simple peak detector would consist of a diode, capacitor and resistor. The diode conducts an AC signal only in one direction. During the negative cycle of the ac signal, the diode is reverse biased thereby ensuring that the signal does not flow through. The capacitor samples the input voltage and stores it long enough before the charge leaks to ground via the R1 resistor.

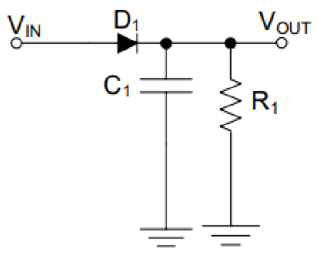


Figure 16: Basic peak detector topology.

The two-stage peak detector topology in Fig. 17 is chosen instead of the basic version shown in Figure 16 because the topology in the basic version would have an output voltage that is the input voltage minus a voltage drop that is equivalent to the drop in voltage across the diode.

$V_{out} = V_{in}-V_{diode}$;

The following is the peak detector circuit topology that was chosen to be implemented in the PCB design. It is a two-stage peak detector topology with both stages amplifier connected in unity gain configuration, R1=R2, R3=R4. The amplifier A01 charges up capacitor to peak value, removing the variability of input impedance. The amplifier A02 acts as output buffer, preventing unintentional discharging of C1 caused by next stage loading impedance. Resistors R2 and R3 limit current into amplifiers A01, A02 respectively by preventing C1 from discharging through them [17].

Most importantly, this topology eliminates the problem with the output voltage having voltage drop due to the diode voltage by including the diode in the op-amp feedback loop. Due to the op-amp's feedback mechanism, it forces the output voltage to be the same as the input in the positive cycle, but during the negative cycle, the presence of the diode at the output restricts the signal from flowing backwards, thereby giving us a circuit that outputs the peak voltage of the input signal.

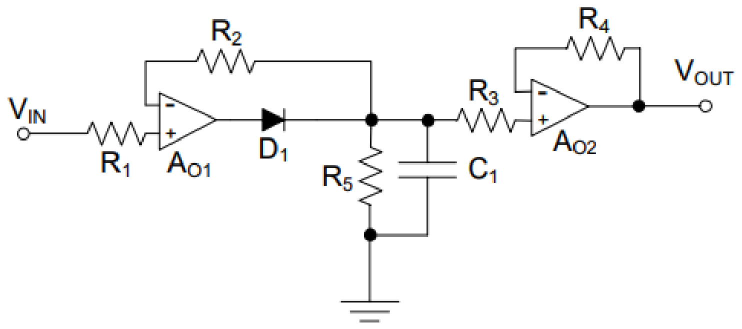


Figure 17: Two-stage peak detector topology.

4.6 Microcontroller (MCU)

The microcontroller of this system plays the role of using its peripherals to enable the running of the system. The peripherals used include Analogue to Digital converter (ADC), two wire interface (I2C), serial communication (UART), GPIOs to interface with the surrounding circuitry. The following table summarizes the microcontroller's function in this system and the peripherals used to perform them:

Peripherals used in Microcontroller	Function performed
ADC, GPIO	Performs sampling of voltage signals using ADC, compares the values to predetermined limits and uses it to control the MUX.
I2C	Sends command to the external clock to change frequencies as needed.
I2C	Sends command to AD5933 to perform impedance analysis sequences. Receives results and does part of the processing of the data that contains impedance information from AD5933.
UART	• Communicates with serial terminal for debugging purposes initially • To send PCB measurements to software for further processing.
SPI	Communicate with BLE module
All peripherals	The microcontroller also ensures that the power of the system is conserved by switching off the external peripherals when not in use.

Table 4: Microcontroller's function in this system and the peripherals used

- I2c

AD5933 and external clock are interfaced with the microcontroller through i2c. Both devices share the same I2C bus as ATmega328P only has one i2c. The microcontroller which acts as a master differentiates between the two devices by the unique 7-bit address that both devices hold.

- GPIO

The 4 to 1 Mux is controlled by the MCU through GPIO pins. By sending the right voltage level as an output to the select pins of the MUX, we are able to control the right feedback resistor connected on the feedback signal path of the op-amp.

- UART

USART of the MCU was initially configured for debugging purposes. A way to readout the values of the registers from the MCU was needed during the development stages of the system. The easiest way to perform debugging was by connecting a FTDI hardware to the RX and TX pin of the MCU and send the variables that we are interested in along that channel and print out the values on a terminal. An alternative method of debugging was initially tried. This method involved no extra hardware as we would be using the programmer itself. However it is needed to change the fuse bit of the MCU from ISP (In-system programming) to debugWIRE mode. In doing so, we would enter debug mode in Atmel studio where we can set breakpoints, step through line by line of the code and observe the change in register values as well as change in variable values from the watch window. However, this is a very lengthy and error prone alternative hence it was no longer pursues as means for debugging. It is said to be error prone because failing to change the mode back to ISP results in irreversible action on MCU as it involves the fuse bit thus the MCU can no longer be used. Since the need for UART communication for debugging purposes was not anticipated, the PCB built did not have the routings for RX, TX pins and did not have the hardware to support it. Hence an empty Arduino board was borrowed from the lab for this purpose. Wires were soldered to connect RX, TX pinout on Arduino to the corresponding pins on the MCU. Wires for VDD and ground were also routed from the PCB to the Arduino as reference. This UART setup coupled with the terminal extension downloaded and installed in Atmel studio worked very well in debugging the system. Towards the later part of the project, USART was also adopted to interface with the software that processes and displays the final output of the system. More on this will be discussed in the next section.

4.7 Software

Since the microcontroller is a 8-bit one, it cannot handle calculations involving numbers that are bigger than 2 bytes. Hence for further processing of values returned by AD5933 to obtain the actual impedance values consisting of both phase and magnitude of the unknown impedance, we need to transfer the values to software for processing. Also, the use of software enables the plotting of Nyquist and Bode diagrams that are key parts of any EIS results. Two suitable platforms along with its mode of interfacing were compared against each other.

The following two software options were contemplated (refer table 5):

Option number	Software platform	Mode of interfacing
1	Android app	Bluetooth Low Energy(BLE)
	Pros: • Attractive solution that is part of Internet of things (IoT) concept whereby information are wireless transferred to the clouds. • User friendly as the controlling and viewing of data can be done using hand-held smart devices	Cons: • BLE based solution require more development stages where by an Android app would be needed to process the hardware results and plot Bode and Nyquist plot. This is not feasible given the time constraint.
2	MATLAB	Serial communication -UART
	Pros: • Simple solution to develop. UART and MATLAB programming are relatively more straight forward. • MATLAB is versatile for graphing and visualising data purposes.	-

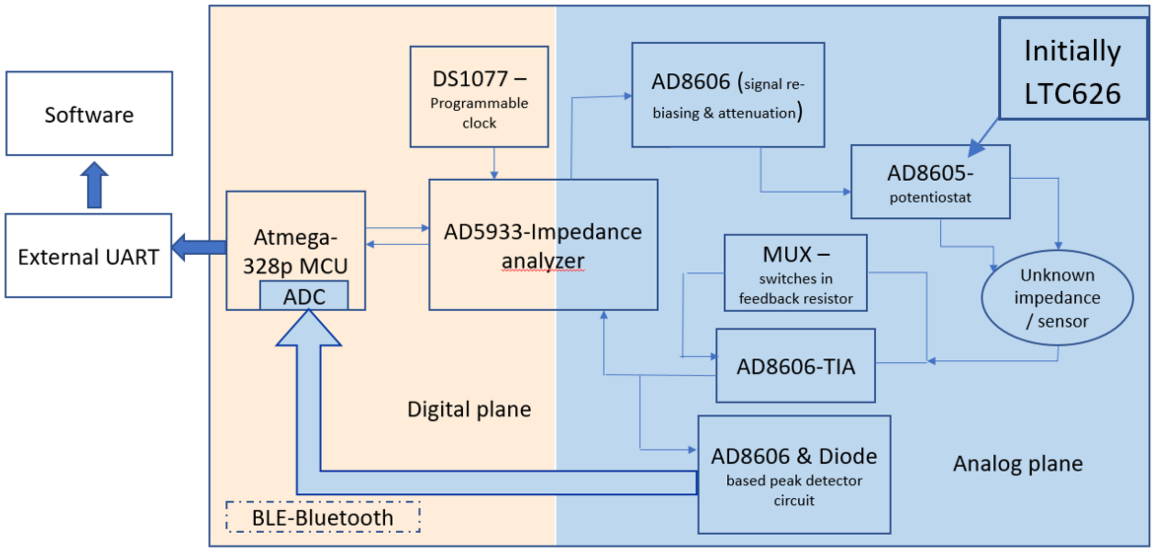
Table 5: Comparison table of software platform and its supporting hardware.

CHAPTER 5

PCB IMPLEMENTATION

5.1 Components Selection

The component diagram below shows the finalized components on the PCB. This section describes the selection of the components PCB and the criteria of choosing them.



Software
External UART
DS1077 – Programmable clock
Atmega-328p MCU
ADC
AD5933-Impedance analyzer
AD8606 (signal re-biasing & attenuation)
Initially LTC626
AD8605-potentiostat
MUX – switches in feedback resistor
Unknown impedance / sensor
AD8606-TIA
Digital plane
AD8606 & Diode based peak detector circuit
Analog plane
BLE-Bluetooth

Figure 18: Component Diagram

<u>Instrumentation components</u>

Instrumentation components consist of amplifiers to form 1st gain stage, control amplifier/potentiostat and transimpedance amplifier (I-V converter). In general for all the devices in this system, an important consideration is that the operating voltage should be in the low range of 0-3.3V. The amplifiers selection was narrowed down to the two choices LTC6268 (as potentiostat) and AD8606 as TIA and inverting summing amplifier. The analysis data referring to the points of consideration of both the op-amps are presented in Table 6. The amplifiers should not have phase reversal as this can introduce phase error along the signal excitation path and receiving path of the unknown impedance. The transmit side amplifier should be able output up to 1mA of current since the unknown impedance range is 10 ohms to 100kohms.

Specs/Op-amp	**LTC6268/LTC6269**	**AD8606**
Operating Voltage (V)	3.1 to 5.5V	2.7 V to 5.5 V
Current Consumption: Active Current (mA) Sleep current (mA)	 16.5 0.39	 1 No data
Input Voltage Range:	0-4.5V	0-5V
Input Offset Current:	+- 6fA to 2pA	0.1pA
Output Current:	135mA	±80 mA
Gain-Bandwidth Product	500MHz (at 100kHz gain=5000)	10MHz (unity gain)
Voltage Noise (nV per square Hz)	4.0nV/√Hz	8nV/√Hz
Slew rate (V/us)	1500V/us or 400 V/µs	5 V/µs
Current Noise (pA per square Hz)	7fA/√Hz	0.01 pA/√Hz
Input Bias Current (nA)	±3fA	0.2 pA
Output Impedance (ohm)	Less than 1 ohm	1 ohm
Phase Distortion	Phase change after 100kHz	No phase reversal

Table 6: Amplifier specification data [18][19]

Points of consideration that is specific for the control amplifier is that it should have low input bias current so that no current flows from the RE electrode such that the I_{RE}=0. The voltage output at the amplifier is around 10mV, hence the voltage noise and current noise needs to be low. Also, its output impedance needs to be very low so that when connected in series to the unknown impedance, it does not result in measurement error of the unknown impedance. Based on the specifications in table 6, amplifier LTC6268 was chosen as the control amplifier for those reasons.   Gain bandwidth is a common consideration for both transmit and receive amplifiers. Initially when the transmit stage op-amp was configured in topology 1, a gain of 200mV/10mV=20 was needed to achieve the 10mV output. The highest operating frequency is 100kHz, hence the gain bandwidth needed is 2MHz. Both amplifiers on the list fit the gain bandwidth needed. An important consideration when measuring low impedances is that the op-amp connected to it needs to be able to produce high output current in the order of milli-Amps that is needed for a load of 10 to 100ohms. This is one of the limitations of using AD5933 as a standalone chip as its internal amplifiers cannot generate the high current output needed for low impedances.

External clock for below 10kHz frequency generation

The external clock is needed to provide clock frequencies between (12MHz to 11kHz) to be supplied to MCLK pin of AD5933. The AD5933 uses this clock input as an oscillator for the in-built DDS to perform frequency sweep from (10 Hz to 100kHz). Another requirement is that the power consumption of this model needs to be low as well as the clock jitter needs to be within 50ppm for a stable signal. The suggestion for a scaled clock option to enable low frequency sweep provided by Analog Devices involved the use of a standard crystal oscillator master clock with the AD9834 chip which acts as a binary clock divider.   It also needs an external high-speed comparator for example ADCMP37x/ADCMP60x at the output to produce the digitally controlled clock [15].

Another suggestion commonly suggested for this kind of application is a programmable oscillator that is simply controlled by the microcontroller using interface like SPI and I2C. Comparison between the two models that represent the suggestions above are detailed in the table below.

Name of component	AD9834 (recommended by AD5933 datasheet)	DS1077LZ-40+ [20]
Type	Complete DDS (with sine output and also square wave as clock)	Oscillator divider
Power	20 mW (with power-down option)	Standby Current during Power-Down Mode: 5 microAmps Supply current active: 15 mA This translates to Power =49mW at 3.3V operation.
Voltage Supply	2.3 V to 5.5 V	2.7V to 3.6V
Output frequency	up to 37.5 MHz	40.0MHz to 4.87kHz
Communication	3-wire SPI interface	2-wire serial interface (SDA, SCL)
Verdict		This is preferred as no external components is needed and it has lower configuring complexity

Table 7: External clock IC/circuit options with configurable output frequency

Based on the table comparison between the two options, the option of using DS1077 was chosen because it no external circuitry is needed for this option as compared to the AD9834 which needs additional components such as an external reference clock plus a high speed comparato. From the power standpoint, since clock circuitry is one of the most power hungry portions of a circuit, choosing option two restricts the power consumption to only one module instead of several modules in option 1. An added advantage is having lesser components decreases the hardware configuring complexity .

Mux selection

The circuit also needs multiplexor to switch in the right feedback resistor based on the range of impedance that it is measuring. Since the ranges are seperated into four where 10Ω-100Ω belongs to range 1, 100Ω to 1k Ω belong to range 2, 1kΩ -10kΩ and 10kΩ to 100kΩ belong to range 3 and range 4 respectively. ADG804 was chosen as it is a 4 to 1 CMOS multiplexer with very low ON resistance and low channel leakage current of 0.01nA. This ensures that the current that flows from the impedance in this system which ranges from 100nA to 1mA is convert well to voltage without any leakage. Another criteria is that the multiplexor needs to be very low power. Figure below summarises the key specification of ADG804 which led to it being chosen.

ADG804

- Device type: Low voltage 4-channel CMOS multiplexer
- 1.65 V to 3.6 V Single Supply
- 0.72 Ω typical - 1 Ω ON Resistance
- 3-bit binary address line
- Analog signal range: 0 to VDD.
- Low channel on leakage current of about 0.1nA typical
- Power requirements: 0.003 uA typically

Figure 19: Multiplxer choice [21]

Microcontroller option

Initially Bluetooth function was very much part of the design hence a microcontroller that support BLE function was favored for the design. Bluetooth is used to wirelessly transfer the AD5933 results to software for further processing during the full cycle of the EIS program as it performs frequency sweep and DFT conversion from (10Hz to 100kHz). Bluetooth low energy (BLE) was chosen as the hardware software interface in the initial design instead of Bluetooth Classic. Bluetooth Classic can handle a much bigger bandwidth of data, but consumes battery life quickly and costs a lot more. BLE is used for applications that do not need to exchange large amounts of data, and can therefore run on battery power for years at a cheaper cost.

For this application, BLE is more suitable since it is only used to transfer AD5933 results which is in the form of a few bytes. Hence the data size and transfer rate is low for this application. Unlike Bluetooth Classic, Bluetooth Low-Energy is usually marketed as a single chip solution commonly known as a System-on-a-Chip (SoC). A BLE chip solution usually includes both the RF transceiver and the microcontroller running the Bluetooth stack (firmware) all in a single chip. Bluetooth Classic on the other hand is usually a two-chip solution: a transceiver chip and a separate microcontroller chip.

ATBTLC1000 Bluetooth SoCs ATBTLC1000-ZR110CA	Nordic’s nRF52832
Power due to BLE usage: – 1.88 µA sleep current – 4.78 mA peak TX current – 5.66 mA peak RX current -15.8 µA average advertisement current	Power due to BLE usage: -5.3 mA peak current in TX -5.4 mA peak current in RX
Power due to MCU usage: -active supply current ranges from 1-4mA -idle supply current ranges from 1-2mA.	Power due to MCU usage: -1.9 µA at 3 V in System ON mode -0.7 µA at 3 V in System OFF mode with full 64 kB RAM retention -1.7 V–3.6 V supply voltage range

Table 8: Comparison of BLE-MCU options in terms of power consumption

After a survey on the latest BLE products, the options were narrowed down to Nordic’s nrf52832 as well as Atmel’s ATBTLC1000 as these two have the lowest power and are the more popular BLE microcontroller solutions relative to the rest. The table in 8 compares both of this options in terms of power consumption.

Option 1 does not come with FLASH memory hence needs external memory to be used in conjuction with this device. Another solution would be to use it as a Bluetooth Low Energy link controller with an external host MCU. Communication between the two happens through UART. The second option however can be readily used if we buy this on a breakout board with readily designed antenna. Sparkfun has a breakout board for this.

Based on the power consumption details in the table above, given that the BLE portion of power consumption is comparable to each other but the MCU in solution 2 appears to outperform solution 1 in terms of power consumption, hence solution 2 was considered.

Further readings on solution 2 did not yield clear information on whether it was possible to switch off the BLE peripheral when not in use as this is biggest source of power dissipation when compared to all the peripherals on the microcontroller. This led to a search for a better option that enables switching BLE module off completely when not in use.

BLE standalone module

One of Nordic's earlier modules is the nRF8001 and it does not come in the same package as a microcontroller. nRF8001 is a single-chip Bluetooth low energy Connectivity IC that consists of a fully compliant Bluetooth low energy v4.0 Radio, Link Layer, and Host stack featuring a simple serial interface that supports a wide range of external application microcontrollers. The current consumption at its peak usage is estimated to be 12.5mA This averages down to 9µA for a 1s connection interval. An attractive feature of it is that it has chip select pins REQ, RDY that enables microcontroller to turn on and off the module, giving an option to save power. The nRF8001 also comes in a breakout board form with minimal additional external components such as 32kHz osciillator , level shifter to enable 5V input (its usage is optional) and the BLE chip itself. Hence the power consumption will be as advertised by the nrf 8001 BLE chip manufacturer.

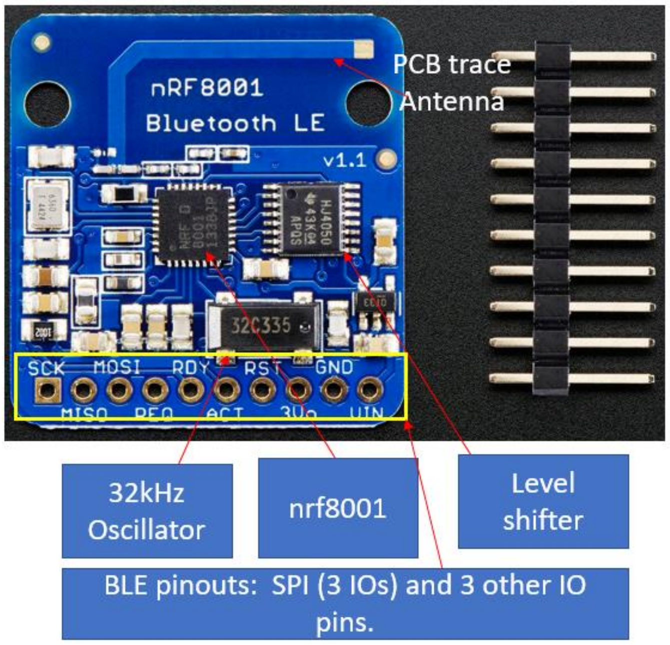


Figure 20: BLE on breakout board

Using nRF8001 in this breakout board form greatly simplifies configuration as it comes with a PCB trace antenna. The pinouts needed for it to be used with a MCU for example SPI pins and 3 other IO pins has been routed out providing ease of use in terms of hardware connection and fast prototyping.

As for the microcontroller, the ATMEGA-328P low power option was chosen. ATmega328P is a low-power CMOS 8-bit microcontroller based on the AVR RISC architecture. It is part of Atmel’s picoPower technology.

The following table summarizes the key specs of this microcontroller.

Operating Voltage	1.8 - 5.5V
Memory	–32KBytes of In-System Self-Programmable Flash program Memory – 1KBytes EEPROM – 2KBytes Internal SRAM
Peripherals	-Two Master/Slave SPI Serial Interface -One Programmable Serial USART (UART) - One Byte-oriented 2-wire Serial Interface (I2C) - 23 Programmable GPIOs
Power consumption (from datasheet as measured at 1MHz, 1.8V, 25C)	-Active mode: 0.2mA -Power-down Mode: 0.1uA -Power-save Mode: 0.75uA (Including 32kHz RTC)

Table 9: Key Specification of chosen Microcontroller [22]

The points taken into account when choosing the right microcontroller module apart from it being low power would be if it has all the peripherals needed to support the functions needed by the components in the system. The peripherals needed for the system are sufficient GPIOs for the MUX, DS1077 clock enabler, ADC, two I2C devices and one SPI for the BLE module.

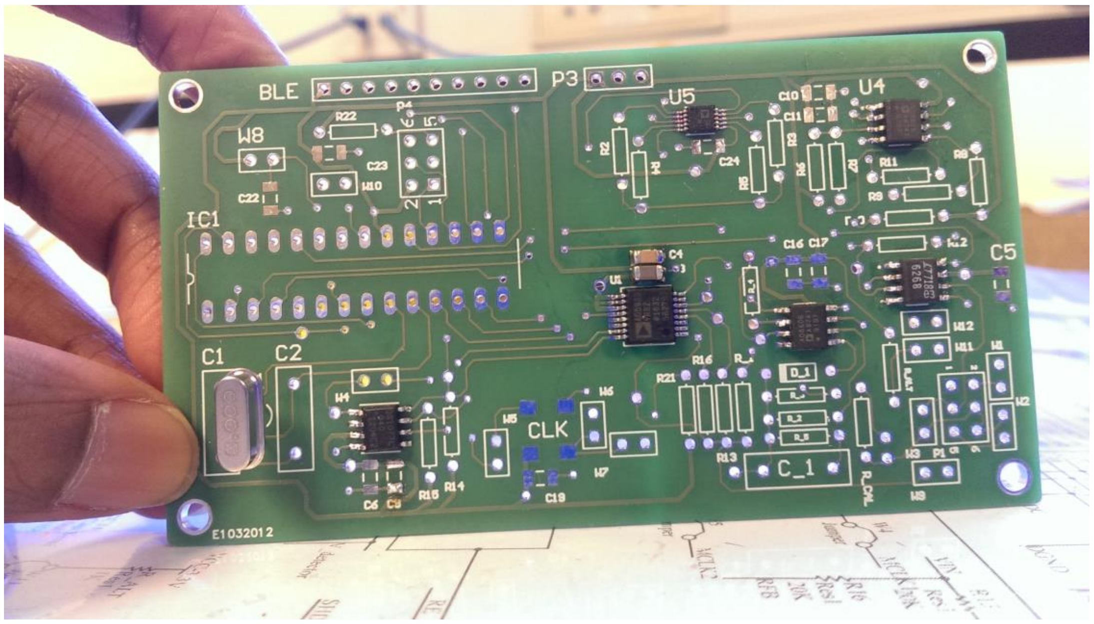
5.2 PCB Assembly and testing

PCB Assembly

The PCB when received were inspected visually for any gross trace misconnections or faults. Subsequently all components were soldered onto it starting with SMD components as they are of lowest height from the surface of PCB. This is followed by components with increasing height such as the pull-up resistors, crystal oscillator, diode and capacitors. Lastly the remaining through- hole components such as the headers as well as pin sockets were soldered in. As the components were soldered in, continuity check was carried out using a digital multimeter to check for shorts caused by excess solder especially between the leads of closely spaced SMD components. Continuity check also ensured there were no open connection due to unsoldered leads. Lastly, continuity check was also carried out between the circuit's main power and ground terminals to ensure there is no shorts between all the power and ground routing of the circuit as this could cause a surge in current potentially damaging the components in the circuit.

Testing of the PCB

The testing of the PCB were carried out individually for every component on the board (refer figure 18). The testing helps us understand each components behaviour and capability thereby ensuring the right hardware configuration is chosen. The components were tested in order of analogue signal based devices first (amplifiers and MUX), mixed signal circuitry (impedance analyser, peak detector & ADC circuit), then digital based devices (programmable clock and BLE module).



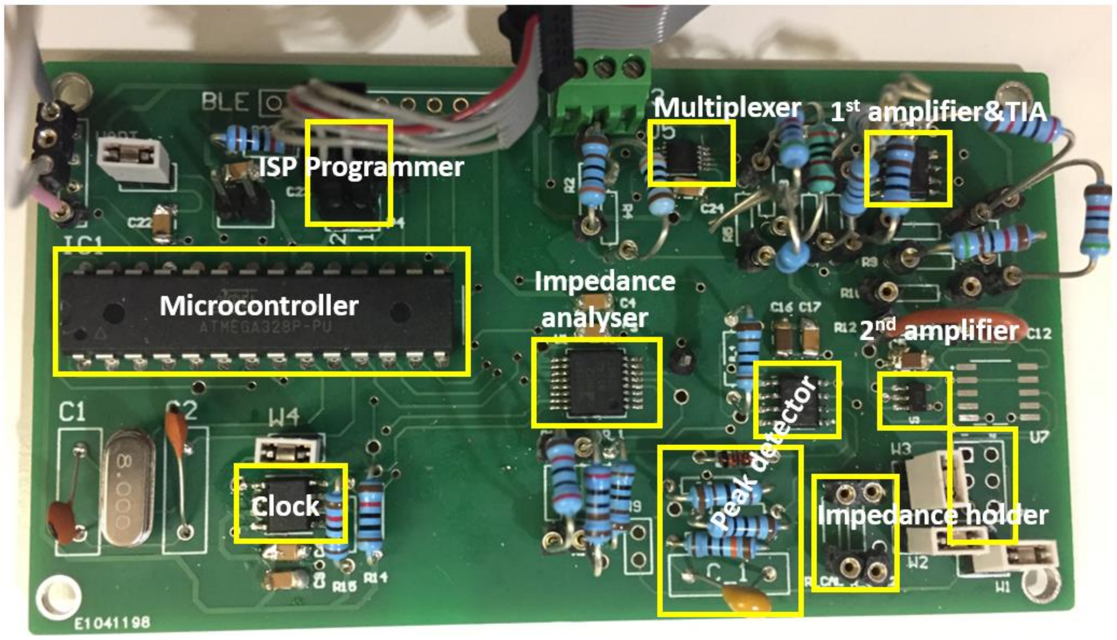
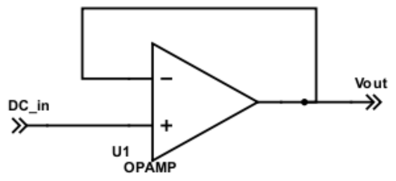


Figure 21: Partially assembled PCB (top); fully assembled and operational PCB (bottom)

<u>Amplifiers testing</u>

Firstly, testing was carried out with the amplifiers AD8606 and LTC which are first stage and second stage amplifiers respectively. The amplifiers were connected in the configuration below (see Fig. 22) with DC input being applied to the positive terminal of the amplifier. A DC sweep between 0 to VDD was applied and the output of the amplifier were simultaneously measured using both DMM and oscilloscope. This testing tells us the voltage swing capabilities of the amplifier. Subsequently the amplifier was connected in the same configuration as before but with a resistor of varying impedance values from 1kohm to 100kohm at the output (see Figure 22 right). This time at a fixed DC input voltage, the output was measured again in a similar fashion. This tells us the capability of the amplifier to source or sink current when connected to a load of varying magnitude. This is important as the impedance specification of this project include impedances as low as 10ohms.



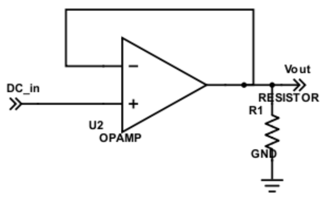


Figure 22 : Amplifier test condition 1 - no load (left), amplifier test condition 2 - with load (right)

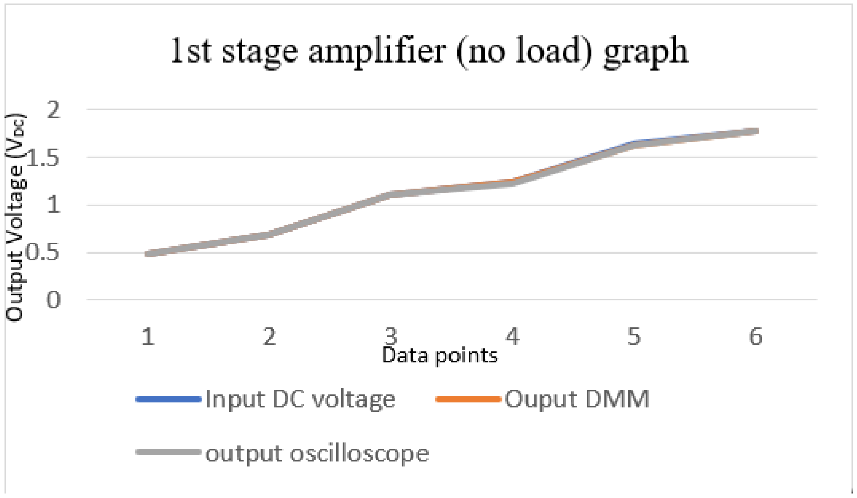


Figure 23: 1st stage amplifier with no load testing results graph

Graph in Figure 23 shows that the amplifier follows the DC input well. From the readings we can observe that the output differs by merely 0.01V. From oscilloscope, the output is observed to be a straight line.

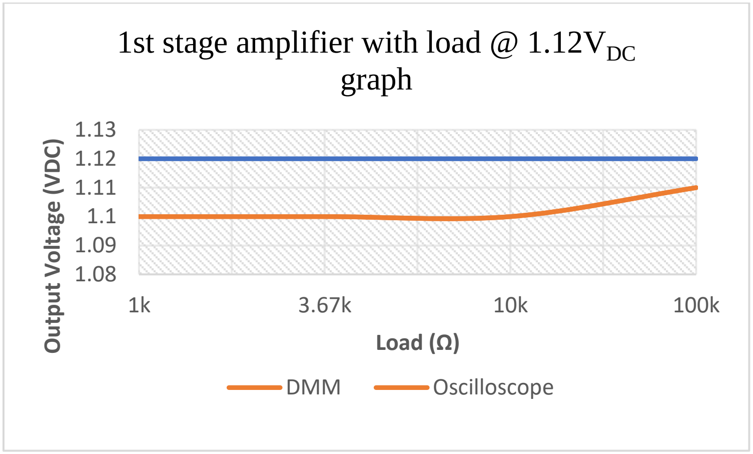


Figure 24: 1st stage amplifier with load at 1.12V$_{DC}$ input testing results graph

Figure 24 shows this amplifier maintains its voltage at 1.10-1.11V which is very close to the input voltage of 1.12V across varying load sizes.

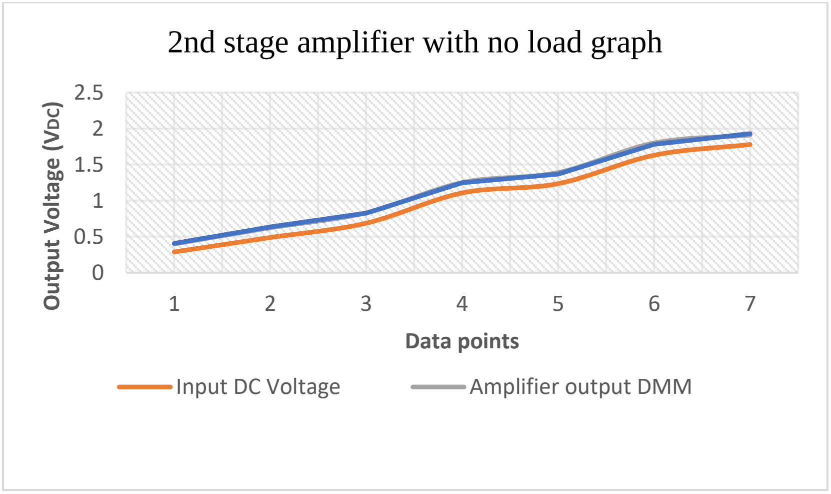


Figure 25: 2nd stage amplifier with no load testing results graph

Graph in figure 25 shows that the readings measured at the output of second stage amplifier are 0.1-0.15 V higher than the input DC voltage when ideally it should be same for a voltage follower configuration.

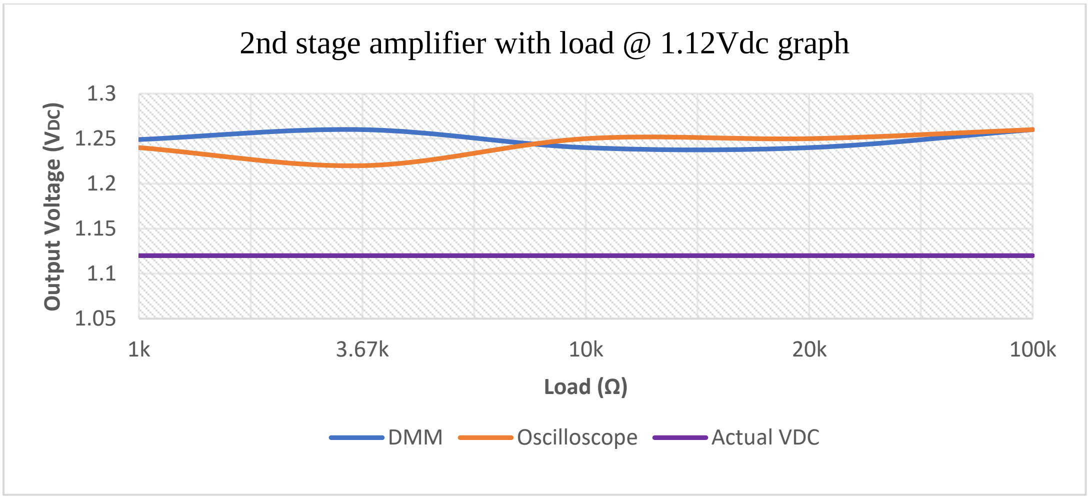


Figure 26: 2nd stage amplifier with load at 1.12V_{DC} input testing results graph

Figure 26 shows voltage at output varies between 1.24V and 1.26V when the output should ideally be same as input voltage at 1.12V across a varying load size of 1k to 100kohms. Oscilloscope measurement at Vout also displays oscillations of 0.1V amplitude instead of flat DC line. This shows that the amplifier is not unity gain stable. Hence it was later replaced with an amplifier similar to the first one (AD8606) as the amplifier needs to be able to operate well in unity gain during calibration and testing purposes.

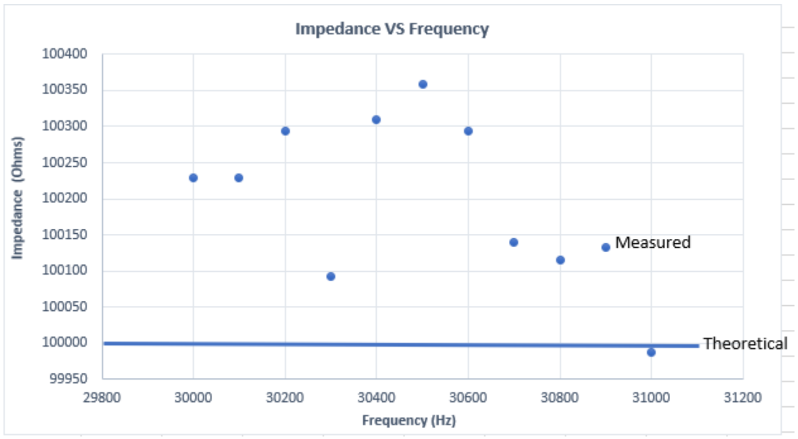
Mixed-signal circuitry testing

Testing of AD5933 was done by writing a code that uses the internal oscillator to perform frequency sweep at 30kHz and analyzing impedance in the form of resistors at range 4 (10kohms-100kohms). The signal path was through the instrumentation in topology 1 but bypassing amplifier 2 since test results indicated that the amplifier is not unity gain stable. The first stage amplifier output had to be connected to the top end of the unknown impedance slot by soldering in a jumper wire to connect both ends.

Initially this setup did not work as the output of the first stage amplifier was still connected to the second stage amplifier via PCB trace. This caused the output at the first amplifier to be stuck at a very high DC offset that could not be reduced by tuning of dc offset providing resistor's value (refer figure 12). This was probably because the second amplifier was still connected at the output and acted as a high impedance load for the first amplifier. As a result, the second stage amplifier had to be de-soldered and removed from the board since it was not possible to bypass it. After doing so, the DC offset at the output of first stage amplifier was successfully set to 1.65V thereby ensuring that there is no difference in DC offset between the two ends of the unknown impedance. This ensures that the current generated and subsequently amplified by the TIA would only consist of AC component and not DC.

During this testing procedure, several points along the signal path were probed to check that the waveform at each point were as expected. Among the points probed were two ends of unknown impedance, AD5933's pinout traces namely VOUT, VIN and RFB (refer final design diagram in figure 39). The waveform at RFB was observed to be abnormal. Further checking with schematic revealed that a trace at the node was wrongly connected. This error was promptly resolved by modifying the resistors connection.

Signals probing after the modification revealed that the signals at all points were as expected. A calibration resistor was first used to obtain the gain factor, the measurement were then repeated with two other resistors 23.9kohms and 100kohms. Both showed good measurement results (refer figure 27) indicating that the hardware setup is workable.



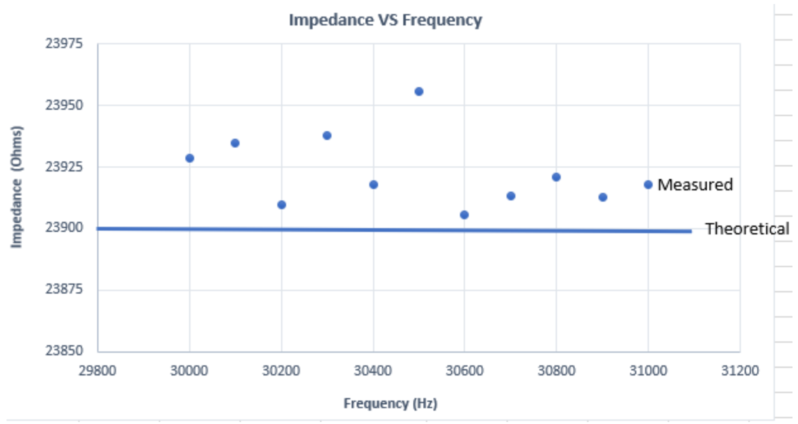


Figure 27: AD5933 measurement results 100kohms (top), 23.9kohms (bottom)

The AD5933 outputs its DFT results as two values in 16-bit two's complement form. Function data_proc (refer figure 28) was written to automatically convert this to signed decimal value using int16_t data type as the sign information needs to be sent to the software to calculate the phase. The phase portion were not calculated at this point since the main concern of testing at this point was to ensure the hardware connections were correct and that there is no major error with the measurement accuracy.

```
int16_t Data_proc(unsigned char data_high, unsigned char data_low)
{
	int16_t data;

	data=(int16_t)data_high*256+data_low;

	if(data > 0x7fff)
	{
		data=0x10000-data;
	}
	return data;
}
```

Figure 28: Code to convert AD5933 2's complement results to signed decimal values

Peak detector circuit testing

In the testing of the peak detector circuit that was built on the PCB, tests were performed to find out the characteristics of the chosen topology and components value. One of the test was to fix the input voltage and sweep the frequency from 10Hz to 100Hz. Based on the results shown in Figure 29, the circuit reflects actual amplitude best at mid-frequency range from 1kHz to 10kHz.

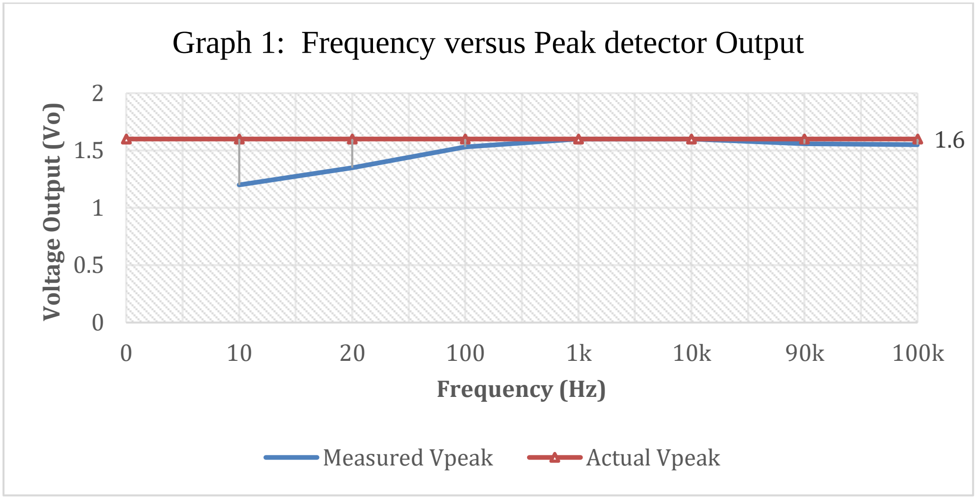


Figure 29: Peak detector output at varying frequencies.

The following figure shows the peak detector circuit output voltage measurement results by fixing the frequency at its optimum operation frequency, and varying the input voltage from 0.3Vpp-3Vpp at a fixed DC offset which is the expected range of operation. The results show that it follows its input quite well.

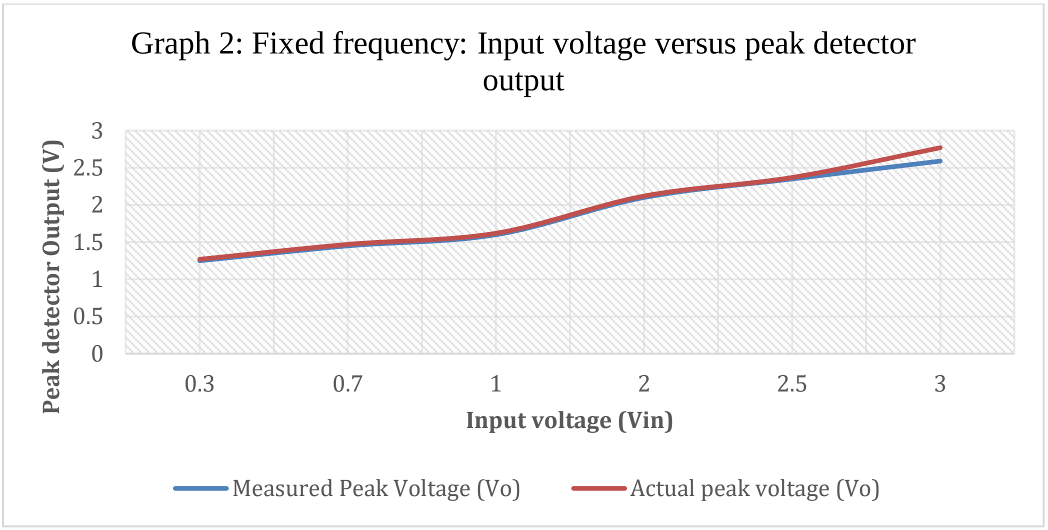


Figure 30: Input voltage versus peak detector output at a fixed frequency

However this peak detector circuit has a certain limitation, it is only able to accurately determine the amplitude of the impedance at limited frequency range hence this becomes a problem when we have an impedance that magnitude which varies in frequency such that its variation exceeds the initial assigned range. For impedance of this kind, the measurement is carried out in a two step process whereby the first measurement is completely done in the range detected initially by the peak detector circuit. After studying the impedance curve produced, in the situation that the impedance exceeds the assigned range, the ADC would saturate and this can be seen in the graph. In this kind of situation, we can repeat the measurement by manually assigning the feedback resistor to the range that is one range below or above it.

The reason for this is that the peak detector circuit only has one set of resistor and capacitor which decides the time taken to discharge based on the following equation [17]:

Where:
$V_{PK(previous)}$ refers to the peak input signal amplitude before the capacitor discharges.
$V_{PK(previous)}$ refers to the peak input signal amplitude after the capacitor discharges.

Hence for example, if the rate of discharge is fixed to accommodate the higher frequencies of 100kHz-30kHz, when it comes to the lower frequencies of 10Hz, the capacitor would discharge at mid signal thus the amplitude of the signal sampled would be rather low and does not reflect the actual signal. The same applies if the R, C values were fixed to accommodate the lower frequencies, when operating at high frequencies the peak detector would take a long time to discharge resulting in it not be able follow the fast-moving signal. In summary, the R, C values can only be chosen to accommodate for one frequency range. And the peak detection is always accurate in the beginning when the capacitor initially charges up. Hence the impedance range estimation in the full EIS flow is always done in the beginning.

Digital circuitry testing

Digital circuitry consists of programmable clock, MCU, and BLE. Testing of the programmable clock was performed by writing the codes based on the datasheet's recommendation. Figure 31 shows the internal block diagram of the clock. The portion of the circuit used to generate the frequencies needed in this project is indicated in the figure.

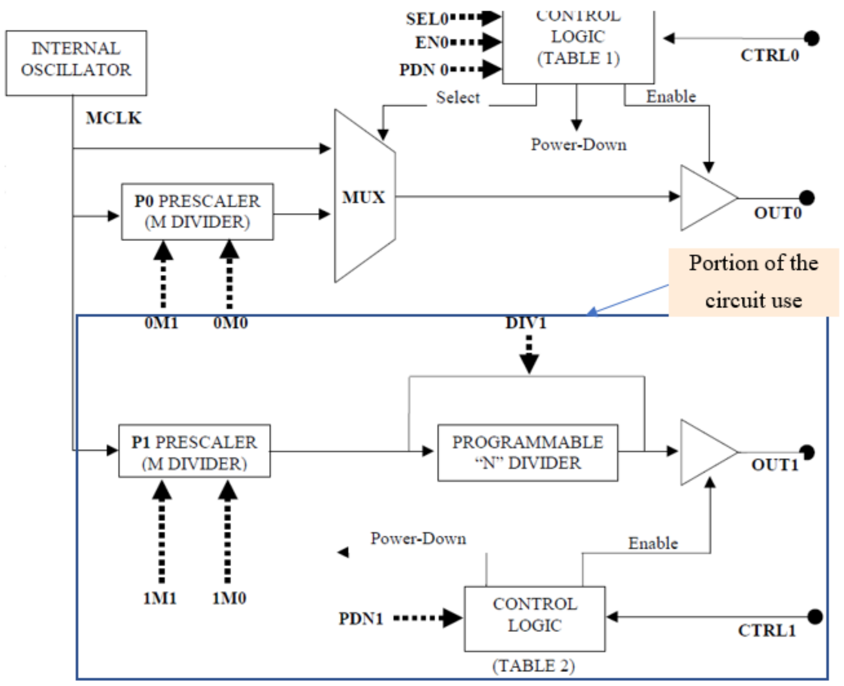


Figure 31: DS1077 partial block diagram [20]

The clock works by scaling master clock 40MHz by factor of 1,2,4,8 via P1 Prescaler block or with additional scaling by "N" Divider block from a factor of 2 to 1024. In order to produce frequencies from 12MHz to 11kHz as the input to impedance analyser's clock pin, we need to scale the master clock using both P1 Prescaler and "N" Divider blocks.

Test codes were written to set P1 Prescaler to 8 and N divider to 63 giving an output of 79.4kHz (40MHz/(8*63)). However, the waveform observed from the scope was at 5MHz. The waveform output indicated a frequency reading that represents the master clock being scaled by the prescaler, PI but not the "N" Divider.

After some troubleshooting, the right waveform at 79.4kHz (refer Figure 32) was obtained by adding a reasonable amount of delay between programming the prescaler and divider commands. This was an important consideration in code of final EIS flow to set the prescaler only once or twice in the beginning and have it immediately followed by a delay.

The clock scaling at subsequent frequency ranges was performed by varying the "N" divider instead of changing both prescaler and divider for every range.

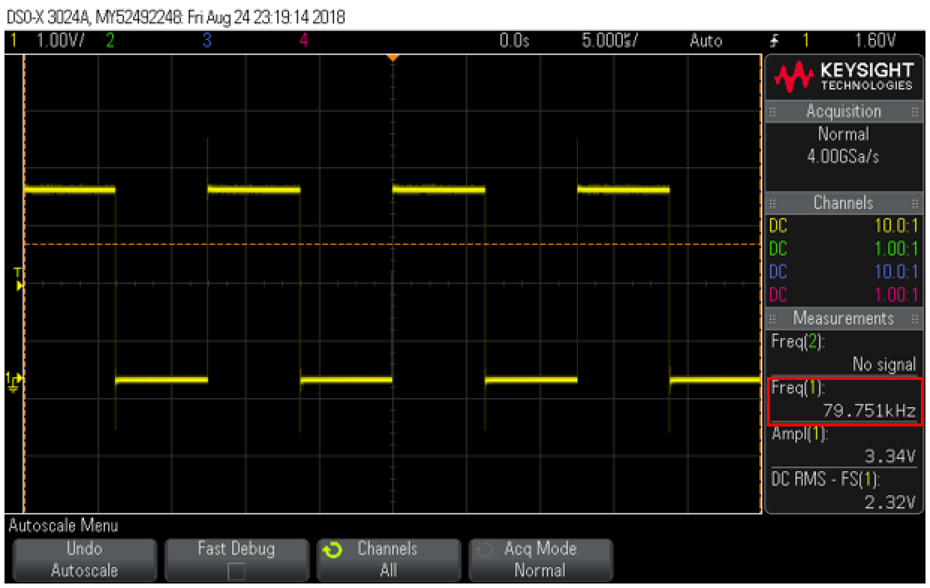
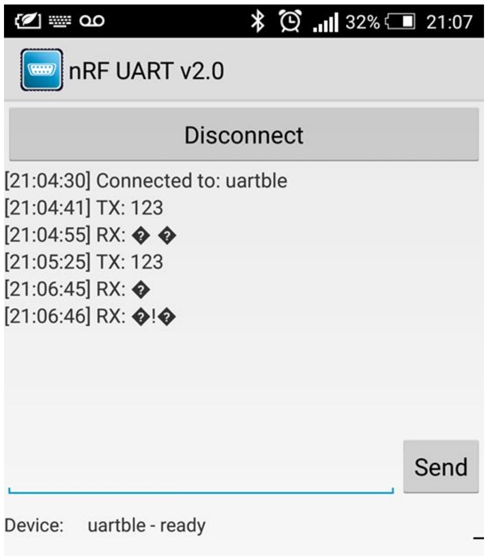


Figure 32: DS1077-40 clock output during testing

Testing of BLE Module

Testing of BLE was also carried out by writing a simple code to configure the SPI and the BLE to send and receive characters on an Android phone app.



```
[21:04:30] Connected to: uartble
[21:04:41] TX: 123
[21:04:55] RX: � �
[21:05:25] TX: 123
[21:06:45] RX: �
[21:06:46] RX: �!�
```

Device: uartble - ready

Figure 33: BLE test results

Based on Figure 33, characters being sent and received were observed but the characters did not have proper ASCII formatting. This confirmed that the hardware connections were correct, but the software code needed troubleshooting. However due to time constraint, it was decided to continue the development with UART/serial communication setup.

Testing of the BLE module marks the end of testing for all the individual components-based testing on the PCB. A new PCB layout that incorporated new footprints for second stage amplifier and other minor changes from PCB 1 was then sent for fabrication. While waiting for the second PCB to be fabricated, the full EIS code was developed and tested using the first PCB. The instrumentation topology 1 was used at this point by configuring it to output an excitation voltage of 200mVpp at 1.65Vdc. When the second PCB arrived from fabrication, it was configured using instrumentation topology 2 to output 10mVpp at 1.65Vdc. PCB 2 with new amplifier in the second stage was tested with the amplifier connected in unity gain configuration. The results are shown as part of the final results in the next chapter.

5.3 Full EIS code development and testing

EIS code development consist of two parts. One of is the microcontroller code and the other is the software portion which covers the processing of the hardware returned results. The hardware results from the PCB is sent to MATLAB for processing. The hardware results consist three types information: 1) the impedance range that is provided by range detecting circuitry, 2) the impedance analyser's real and imaginary registers' values, 3) the current evaluating frequency.

The microcontroller algorithm would be discussed first followed by the software portion. The flowchart for the main program is shown in the figure 34 and 35. The main program flowchart consists of mainly three parts (as indicated with the numbers in the box), namely:

1. Initialization of all components, microcontroller peripherals as well as counter variables. Calling the MCLK programming function whenever the frequency range changes.
2. Acquiring ADC readings, do some processing then return impedance range value. Configuring impedance analyser's settings to prepare for frequency sweep.
3. Retrieving results from impedance analyser chip and printing it to terminal using USART along with the current evaluating frequency.

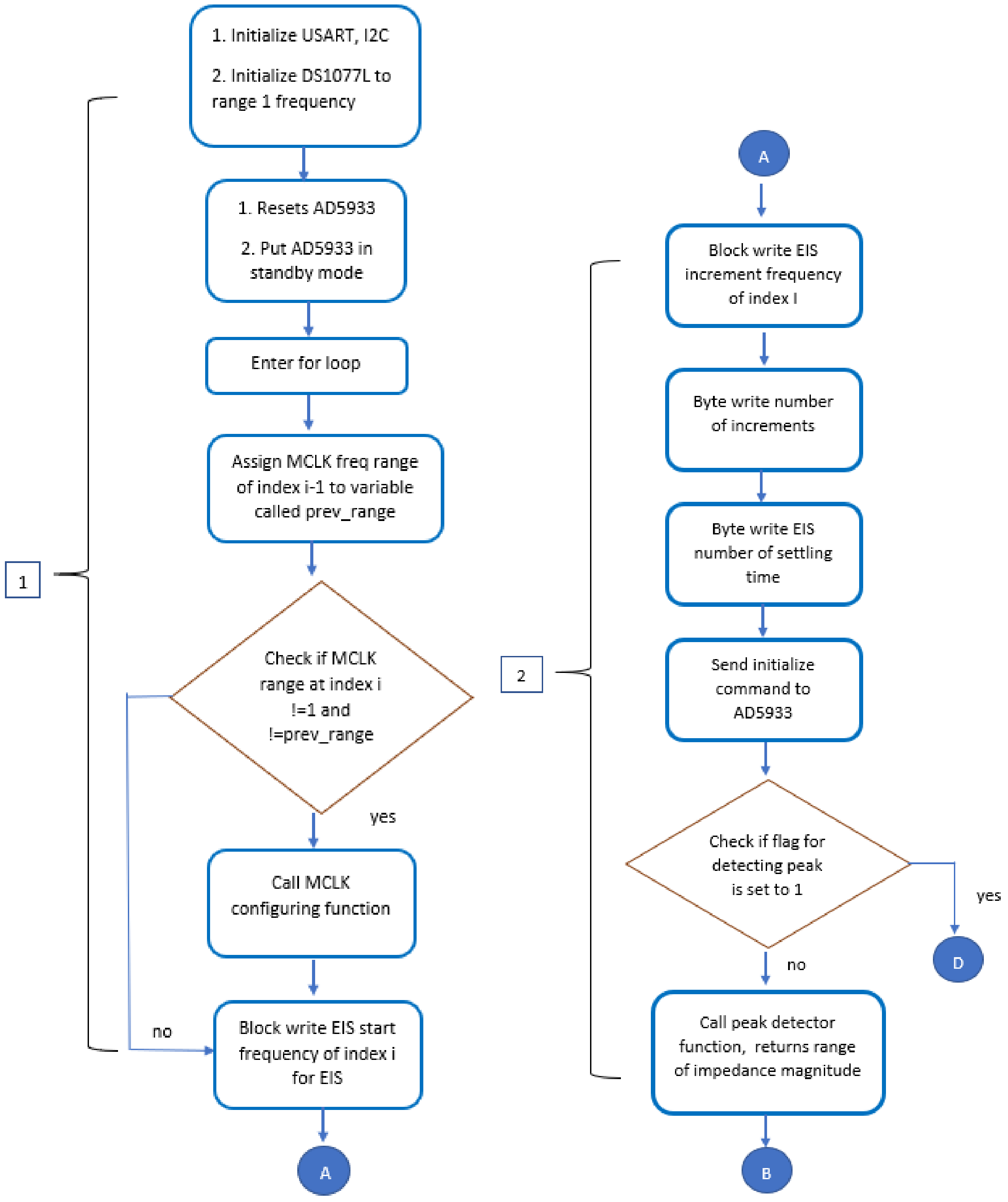

1. Initialize USART, I2C
2. Initialize DS1077L to range 1 frequency
1. Resets AD5933
2. Put AD5933 in standby mode
Enter for loop
Assign MCLK freq range of index i-1 to variable called prev_range
Check if MCLK range at index i !=1 and !=prev_range
yes
no
Call MCLK configuring function
Block write EIS start frequency of index i for EIS
A
1
2
A
Block write EIS increment frequency of index I
Byte write number of increments
Byte write EIS number of settling time
Send initialize command to AD5933
Check if flag for detecting peak is set to 1
yes
no
D
Call peak detector function, returns range of impedance magnitude
B

Figure 34: Hardware EIS main program flowchart part 1

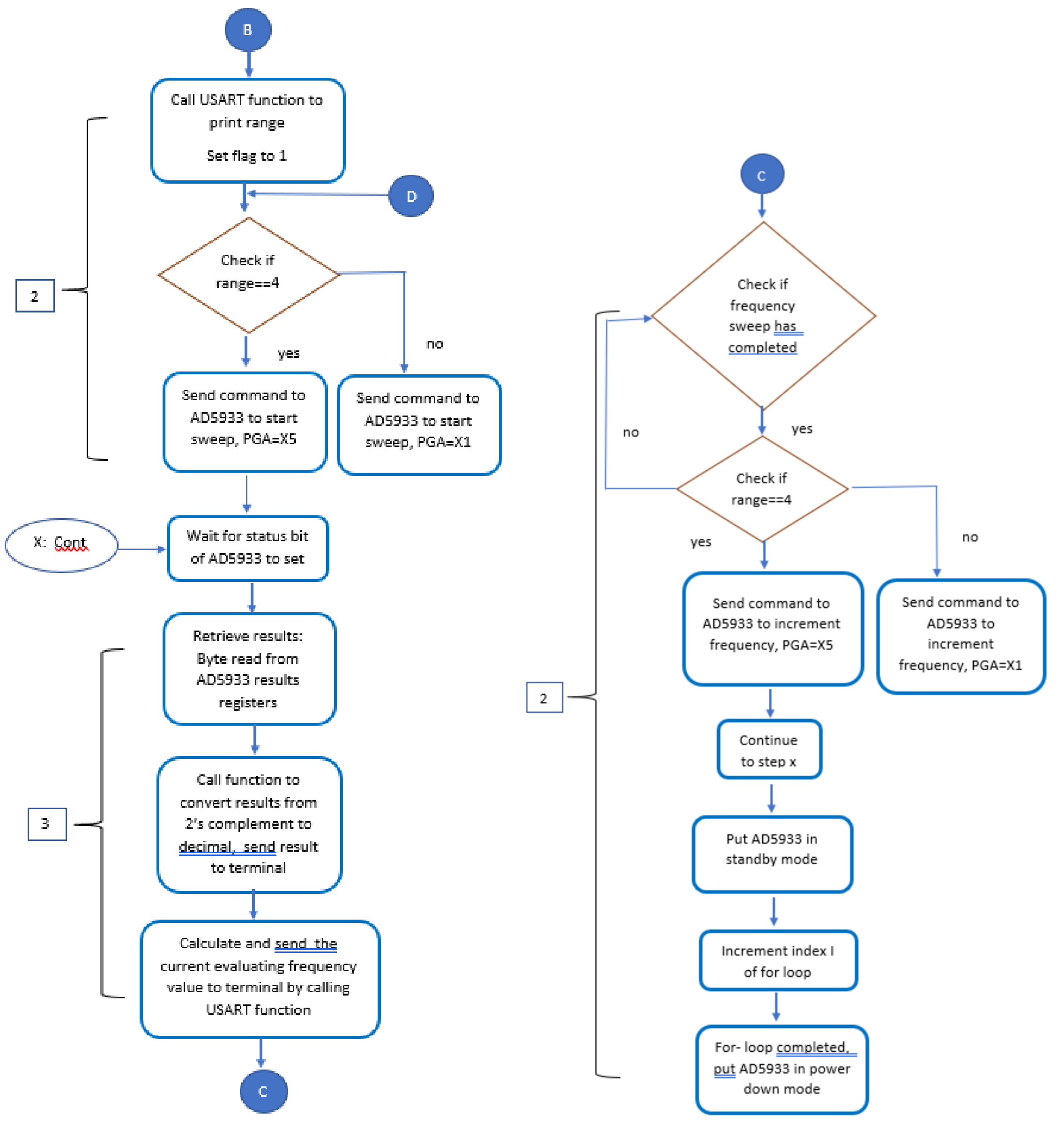

B
Call USART function to print range
Set flag to 1
D
Check if range==4
yes
no
Send command to AD5933 to start sweep, PGA=X5
Send command to AD5933 to start sweep, PGA=X1
2
X: Cont
Wait for status bit of AD5933 to set
Retrieve results: Byte read from AD5933 results registers
3
Call function to convert results from 2's complement to decimal, send result to terminal
Calculate and send the current evaluating frequency value to terminal by calling USART function
C
C
Check if frequency sweep has completed
no
yes
Check if range==4
yes
no
Send command to AD5933 to increment frequency, PGA=X5
Send command to AD5933 to increment frequency, PGA=X1
2
Continue to step x
Put AD5933 in standby mode
Increment index I of for loop
For- loop completed, put AD5933 in power down mode

Figure 35: Hardware EIS main program flowchart part 2

Programming for the start frequencies and increment frequencies at a scaled clock frequency requires some calculation. Table 10 shows the frequency sweep ranges and MCLK clock input frequencies needed to obtain the desired frequency sweep.

The table details the Divider and Prescaler values of the programmable clock used to obtain the final design, the start frequency and increment frequency calculations.

Frequency range /sweep range (Hz)	Frequency of MCLK, f_{clk} (Hz)	Factor, $f_{code} = 2^{27} x(4/ f_{clk})$	Programmed Start frequency (Hz)	Hex code for start frequency (start freq* f_{code})	Programmed Frequency increment (Hz)	Hex code for frequency increment (increment freq* f_{code})	Programmed Number of increments
10k-100k	8M (P=1; D=5)	67.1	90k 70k 10k 30k 9k	5C 25 D8 47 AB A8 0A 3D 18 1E B7 48 09 36 FC	100 100 10 2 10	00 1A 36 00 1A 36 00 02 9F 00 00 86 00 02 9F	9 9 9 9 9
1k-5k	2.5M (P=8; D=2)	214.748	5k 3k	10 62 4E 09 D4 94	10 5	00 08 64 00 04 32	9 9
500-1000	0.8333M (P=8; D=6)	644.27	900 700 500	08 D9 03 06 E1 AD 04 EA 57	10 2 5	00 19 2B 00 05 08 00 0C 95	9 9 9
100-500	79.365k (P=8; D=63)	6.7646k	100 10	0A 52 6C 01 08 3E	5 1 2	00 84 1F 00 1A 6D 00 34 d9	9 9 9
10-100	11.99k (P=8;D=417)	44.776k	10	06 D5 10	1	00 AE E8	9
1-10 (not tested)	8k (P=8;D=625)	67.109k	1		0.1		9

Table 10: EIS Frequency sweep ranges and MCLK clock input frequencies

The start frequency and increment frequency are both 24-bit words (three registers for each frequency) that are calculated using the MCLK frequency as shown in column 3 table 10. Since the 8-bit microcontroller on-board cannot perform calculations of numbers beyond two bytes, the start frequency and increment frequencies for all ranges is pre-calculated and stored as 2-D array in a header file. This is also more efficient as microcontroller resources are not used at every measurement to perform these repetitive calculations. Figure 36 shows the storing of the pre-calculated start frequencies into 2D array.

```
static const uint8_t start_freq[12][3]={{0x5c, 0x25, 0xd8}, {0x47, 0xab, 0xa8},\
{0x0a,0x3d, 0x18}, {0x1e,0xb7, 0x48},\
{0x09,0x36,0xfc}, {0x10,0x62,0x4e},{0x09, 0xd4, 0x94},\
{0x08,0xd9,0x03}, {0x06, 0xe1, 0xad}, {0x04, 0xea, 0x57},{0x0a, 0x52, 0x6c},\
{0x06,0xd5,0x10}};
```

Figure 36: 2-D array used to store start frequencies

```
f_arr1[0]=start_freq[i][j];  f_arr1[1]=start_freq[i][j+1];
f_arr1[2]=start_freq[i][j+2];


Block_write(Freq_high, 3, f_arr1);

uint8_t  Block_write(uint8_t reg_loc, uint8_t byte_num, uint8_t data_p[3])
{
       uint8_t i;
       char y[20];
       Set_pointer(reg_loc);//set the pointer location
       Send_start();
       Send_adr(SLA_W);
       Send_byte(0xa0);     //Block write command code '1010 0000'
       Send_byte(byte_num);

       for(i = 0;i < byte_num;i++)  //Send the data bytes
       {
              Send_byte(data_p[i]);
       }
       Send_stop();
       return 1;
}
```

Figure 37: Code to send start frequency from 2D array to on-board RAM using block write

Figure 37 shows the 2-D array values being passed to an array then written to AD5933's on-board RAM using Block write. The same method was followed for increment frequencies. Using Block write is more advantageous as it is faster for writing to three registers at a single time than sending data in bytes.

As can be seen in table 10, multiple start frequencies share the same MCLK frequency hence care is taken in the code (please refer to flowchart figure 34 left) not to program the clock repeatedly with same conditions as it is time-consuming and not efficient. This is done by a simple condition checking to see if the previous loop index shares the same MCLK frequency range as the current one.

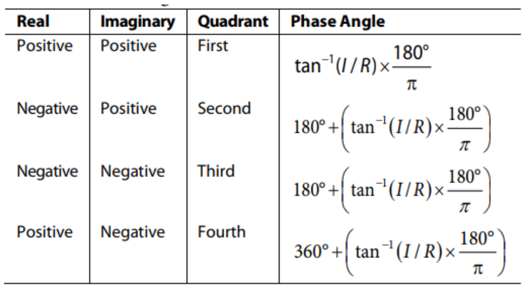
In the microcontroller code, switch-case statement is used to assign the right divider and prescaler values to the DS1077 clock. Switch-case statement is a preferred way of dividing codes based on conditions because it is much faster as it does not check for every condition as in an if-else statement based condition checking. Switch-case statement was also used in assigning the right MUX output in the impedance range detection algorithm.

For impedance magnitude that belong to range 4 (10kohms to 100kohms), the gain required at the TIA is the highest hence it is divided into two stages, one is from the feedback resistor itself, the other is from the impedance analyser's internal PGA (Programmable gain Amplifier). Referring to the flowchart in Figure 35, the code checks to see if the range is of range 4 and assigns the appropriate PGA setting.

The software portion of the algorithm in MATLAB consist of a calibration script and PCB results data processing script. Calibration script accepts the same set of arguments mentioned above from the hardware and generates gain factor and system phase for all the measurement frequencies. There are four sets of gain factor and system phase matrixes in MATLAB corresponding to the four impedance ranges. All these are pre-calibrated values that stored in MATLAB. Since there is no need for change in hardware settings, calibration does not take place for every measurement. System phase is calculated based on the equation 3 and adjustments are made to the inverse tangent result based on the sign of the real and imaginary values.

Measured Phase = $\tan^{-1}$(I/R); Equation 3

Figure 38 shows angle calculation based on the sign of real and imaginary values that indicate the quadrant in which the angle lies in.



Real	Imaginary	Quadrant	Phase Angle
Positive	Positive	First	$\tan^{-1}(I/R)\times\frac{180°}{\pi}$
Negative	Positive	Second	$180°+\left(\tan^{-1}(I/R)\times\frac{180°}{\pi}\right)$
Negative	Negative	Third	$180°+\left(\tan^{-1}(I/R)\times\frac{180°}{\pi}\right)$
Positive	Negative	Fourth	$360°+\left(\tan^{-1}(I/R)\times\frac{180°}{\pi}\right)$

Figure 38: Phase angle adjustment based on quadrant [14].

Data processing of PCB results script in MATLAB can be summarized in the following manner:

1. Reads in PCB results consisting of impedance range value, AD5933 real, R and imaginary, I results and evaluating frequency and stores them in a matrix A.
2. Extracts range value and stores in variable.
3. Converts matrix A into a three-column matrix whereby the individual columns are real, R, imaginary, I and frequency.
4. Checks the range values and uses the corresponding gain factor for impedance magnitude calculation.

   Magnitude without calibration=sqrt(R^2+I^2);

5. Phase is calculated in a similar manner as mentioned above using R, I values and its sign information followed by the formula below, where the unknown impedance phase is:

   Phase_unknown=Measured_phase (after angle adjustment)- system phase.

6. For phase values above 180 and below -180 degrees, the values are adjusted by subtracting 360 and adding 360 respectively to make sure the values are within two quadrants.

7. Subsequently, the calculated impedance magnitude, phase and frequency values are re-arranged in increasing order and plotted in the form of Bode and Nyquist plots.

CHAPTER 6

RESULTS AND DISCUSSION

This chapter presents the Final Design board measurements and full EIS testing results RC network-based impedances. Discussion is provided on the issues affecting accuracy and the implemented solutions to overcome those issues.

6.1 Board measurements

This section details the board measurements of the final design as shown in Figure 39 below.

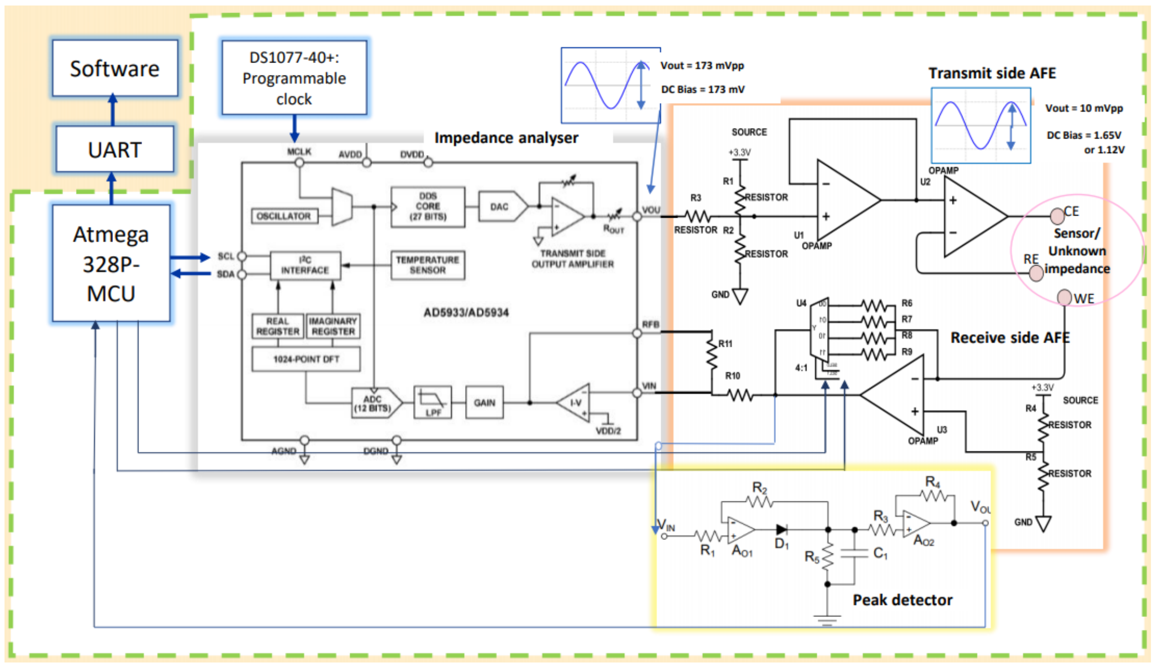
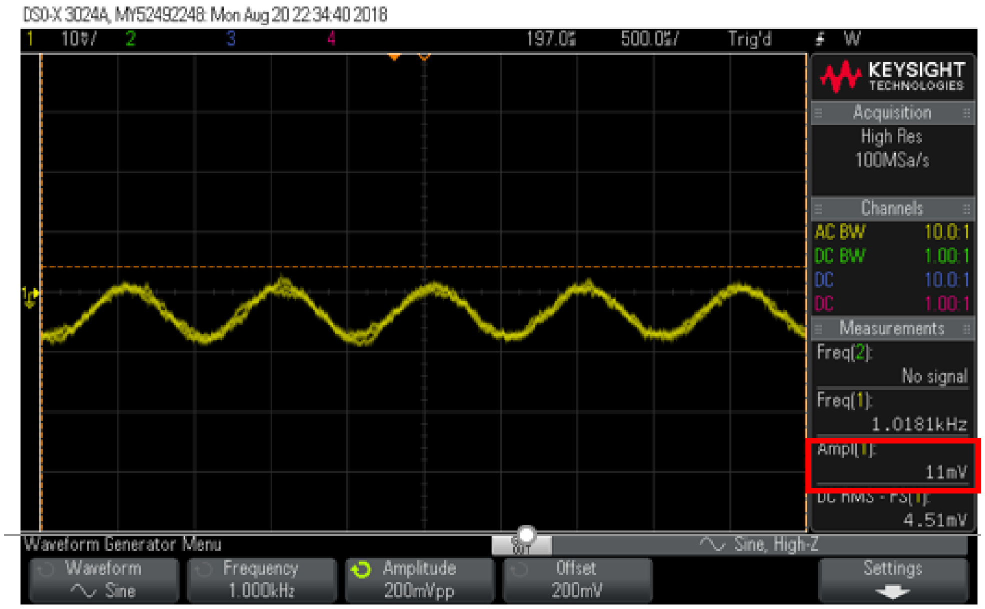


Figure 39: Final Design of full system

These results were obtained while testing out the full EIS measurement algorithm. Figure 40 shows board measurement taken at output of second stage amplifier in the transmit side. This millivolt ac signal which travels on a high DC value due to the circuit biasing requirements was measured using the oscilloscope by enabling the AC-coupled setting.

Waveform output at this point indicate that the designed topology (topology 2 from Figure 13 in Chapter 4) is successful in producing the desired 10mVpp waveform at 1.65Vdc.


DSO-X 3024A, MY52492248: Mon Aug 20 22:34:40 2018

1 10mV/ 2 3 4 197.0µs 500.0µs/ Trig'd W

Acquisition
High Res
100MSa/s

Channels
AC BW 10.0:1
DC BW 1.00:1
DC 10.0:1
DC 1.00:1

Measurements
Freq(2): No signal
Freq(1): 1.0181kHz
Ampl(1): 11mV
DC RMS - FS(1): 4.51mV

Waveform Generator Menu — Sine, High-Z
Waveform: Sine
Frequency: 1.000kHz
Amplitude: 200mVpp
Offset: 200mV
Settings

Figure 40: 10mVpp output waveform

The measurement in figure 41 was taken at the receive end of the AFE before the signal enters impedance analyzer chip (see Figure 39) while exercising the code to perform frequency sweep from 10Hz to 100Hz. The measurement shows that the programmed settings (shown previously in table 10) for both the programmable clock and impedance analyser chip is successful in extending the range of frequency down to 10Hz.

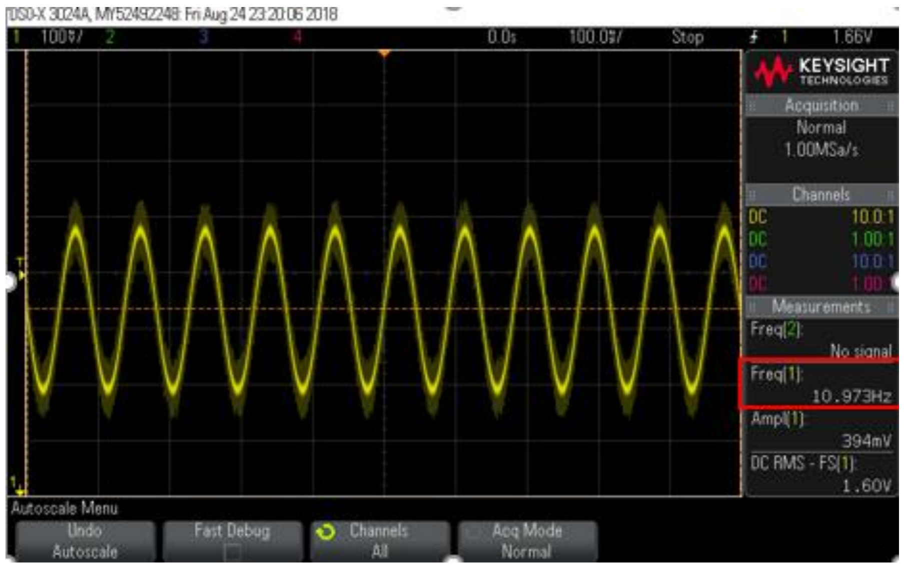


Figure 41: Board measurements results at 10Hz

Waveform in Figure 42 is measured at the same point as before but this time while looping the code to perform frequency sweep starting at frequency of 90kHz. The waveform shows a measurement of 89kHz which indicates that the chosen settings are again able to produce the required frequency.

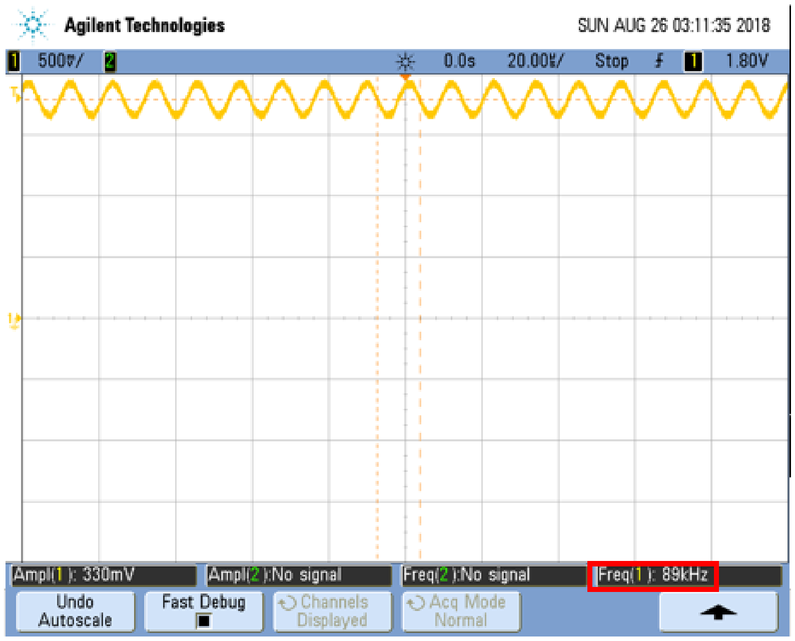


Figure 42: Board measurements 90kHz

Figure 43 shows the measurement of 10Ω resistor. The measured impedance values vary between 10.2ohms to 11ohms. This proves that the front-end amplifier chosen is able to produce the required current for low impedances measurements.

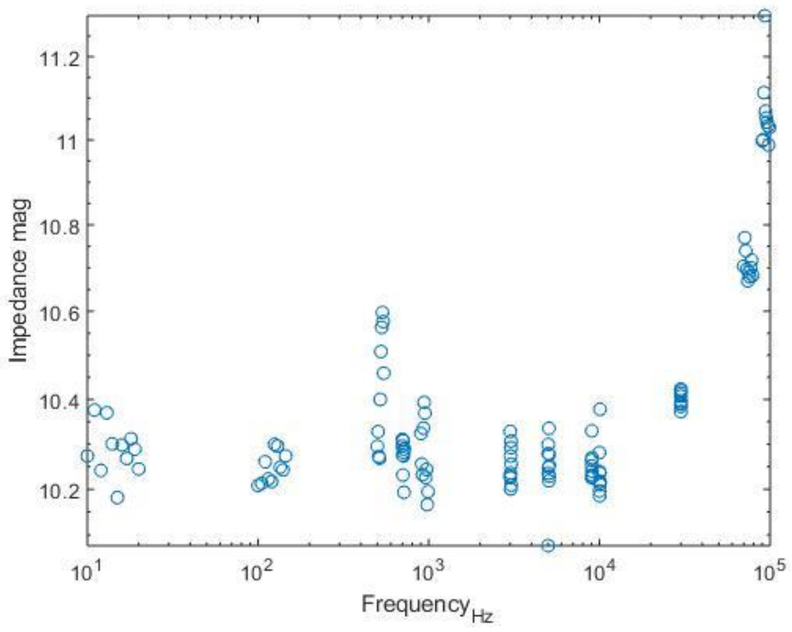
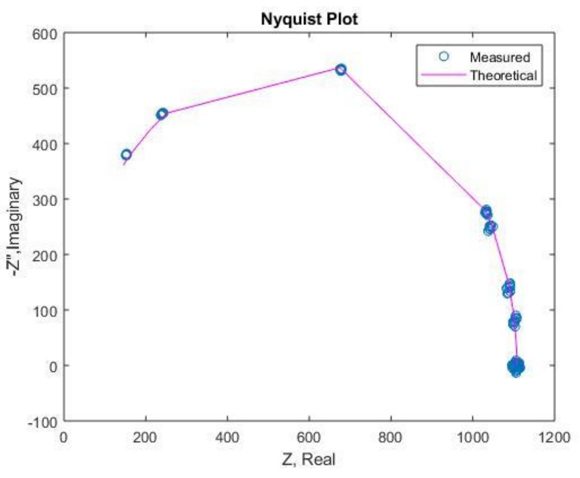


Figure 43: 10 ohms impedance measurement

6.2 Final Results plots

This section shows the plots for the final results. These plots represent the impedance analysis of a Simplified Randles circuit with component values of (Rr=1.1kΩ, Cdl=3.9nF and Rs=10Ω) (refer to figure 44 for circuit configuration). The Bode and Nyquist plots of the measurement results were generated using MATLAB as part of the software processing steps mentioned above. From the plots, we can observe that the results are quite accurate and overlaps on the theoretically calculated values. The theoretical values were calculated using Simplified Randles Cell equations detailed in Chapter 2.



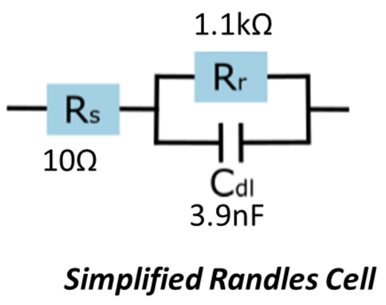
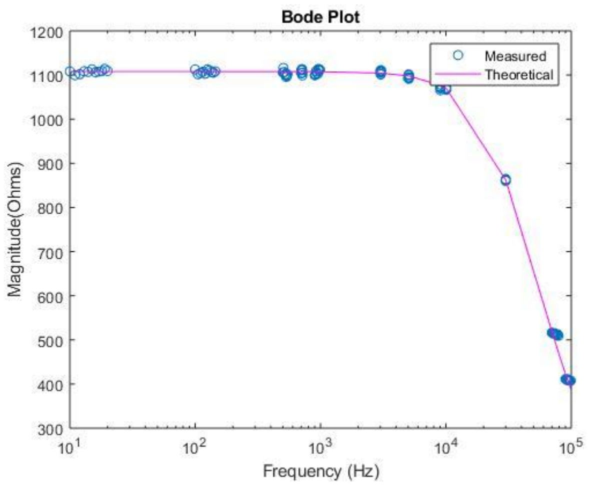


Figure 44: Nyquist plot (left), Simplified Randles circuit with RC component values measured to obtain this plot.



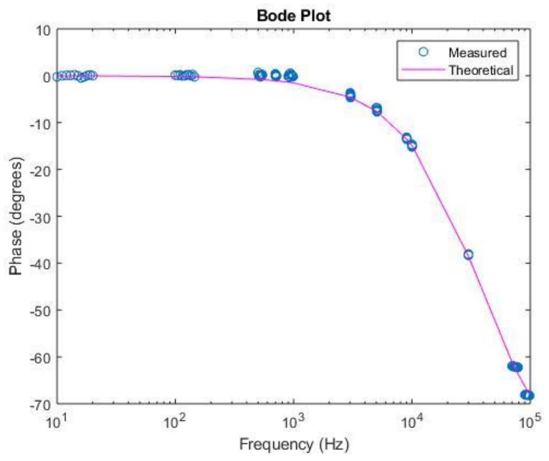


Figure 45: Bode plot of the same measurements

6.3 Accuracy improvements to achieve final design

This section describes the initial errors encountered when implementing the full EIS code and the improvements performed to achieve the final design's accuracy. A brief overview on the errors observed would be given through the graphical plots of measurement data. A comprehensive discussion on the source of these errors and improvements made is detailed in the next section.

Figure 44 shows an example of error observed in two different frequency ranges when performing the full EIS measurements. Spikes are seen at 10-120Hz range frequency giving an error of 200ohms when compared to theoretical value.

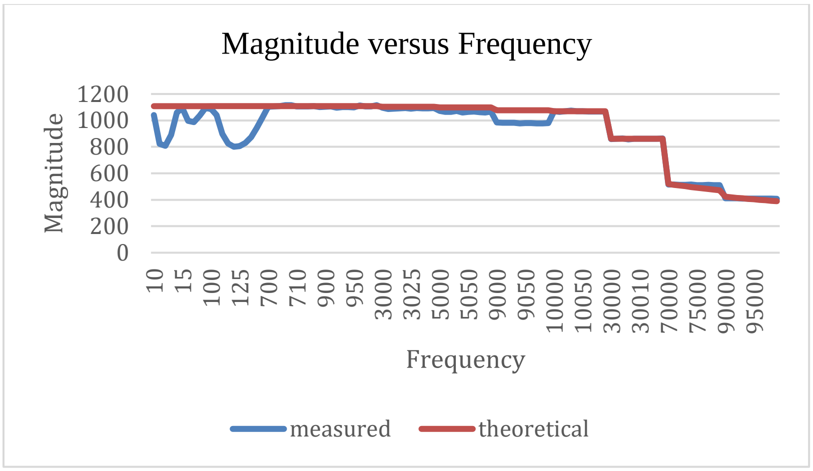


Figure 46: Errors in low frequency range

The figure below shows graph of the same circuit's measurement results but after revising the clock input frequency, MCLK to the impedance analyser chip and lowering i2c clock signal frequency. The spikes are no longer present, and measurement results is similar to theoretical values for this range. Further details on the effect of both frequencies design choices on measurement accuracy will be explained in the sections to come. However, there is still have some error in the range of 3kHz to 9kHz and it is very much more obvious in phase plot as seen in the next figure, Figure 47. Again the MCLK frequency for this range was re-examined, it was understood that setting the MCLK frequency at 1MHz to perform impedance analysis at 9kHz was not suitable.

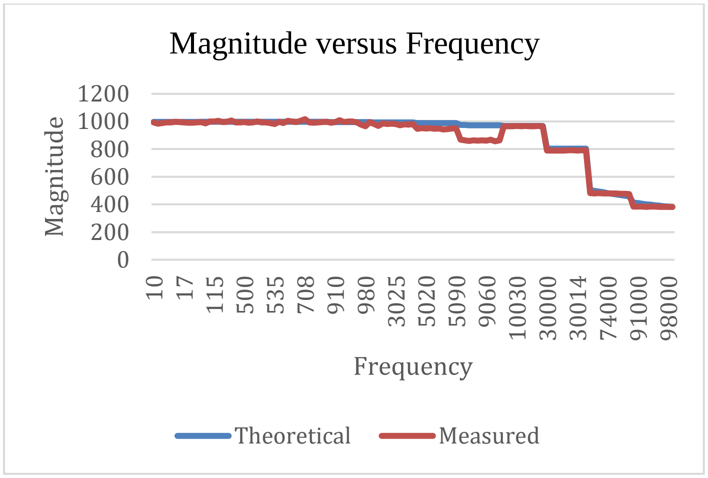


Figure 47: Improved low frequency measurement but with remaining error at 3k to 9kHz magnitude plot

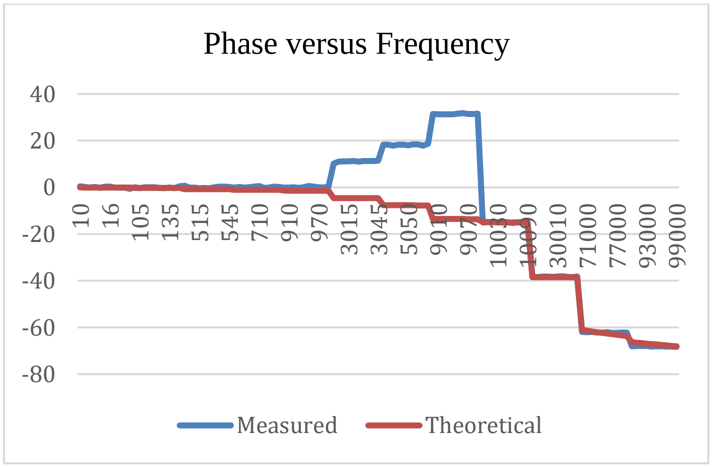
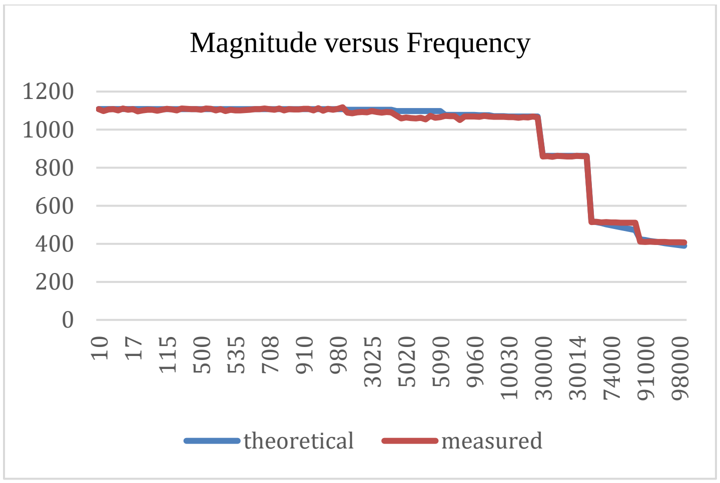


Figure 48: Error at 3k to 9kHz phase plot

After shifting the 9kHz analysis to a MCLK frequency that is higher, from 1MHz to 8MHz, a noticeable improvement in seen for this range, refer magnitude and phase plot of figure 49.



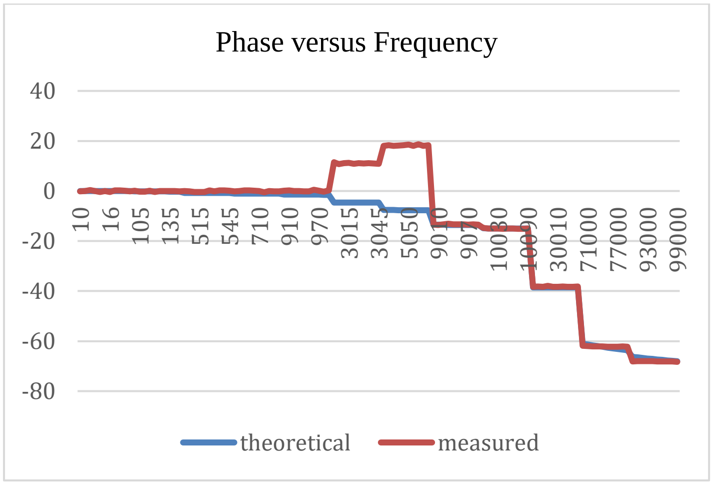
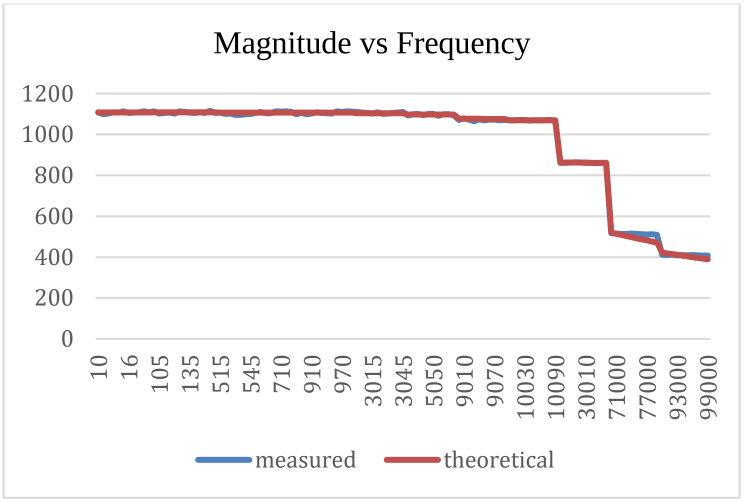


Figure 49: Error in 3kHz to 5kHz range but 9kHz range improved.

From the two plots above, it is clear that we are left with two frequency ranges that still displayed some error namely the 3000-5000Hz frequency range which still had an MCLK frequency of 1MHz. Again the MCLK frequency was increased, but this time to 2.5MHz to perform analysis on this range. Graphs in Figure 50 display the results after this improvement, a much better measurement result that is comparable to the theoretically calculated values was obtained.



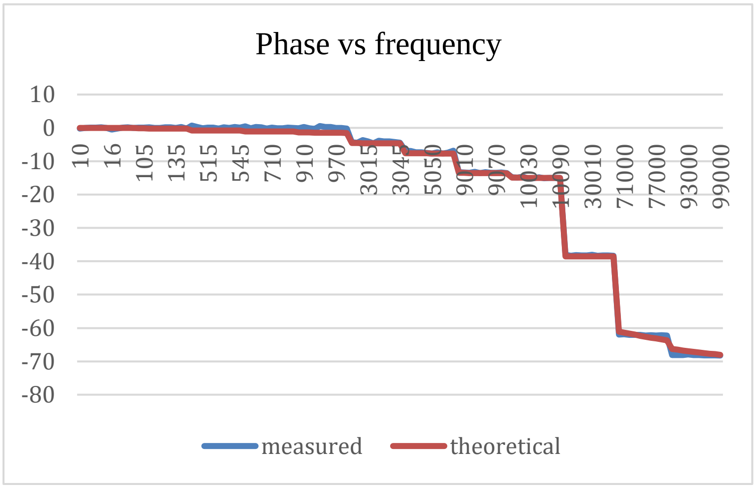


Figure 50: Graph of magnitude and phase after all improvements

In a nutshell, we have observed that the right MCLK frequency (clock scaling) and I2C frequency plays a role in obtaining measurements with good accuracy.

6.4 Discussion on factors affecting measurement accuracy

The main factors that influence accuracy in this system are identified to be the choice of MCLK frequency when applying scaling of clock to extend measurement range, the I2C clock frequency when analysing impedances at low frequencies and lastly is the noise arising from biasing using signal from on-board power supply.

1) Choice of MCLK frequency and its relation to the accuracy of the DFT calculation.

The wrong choice of MCLK frequency leads to poor impedance measurement results. The impedance analysis operation in AD5933 involves the ADC sampling and converting 1024 points (N = 1024) then providing these samples to the MAC unit to perform the DFT. The sampling frequency of the ADC, f_s is as given below:

$$f_s = \frac{f_{MCLK}}{16}$$

The resolution of the DFT is given by:

$$f_{resolution} = \frac{f_s}{1024}$$

At a MCLK input frequency of 16MHz, the resolution of the DFT engine is approximately 1kHz, meaning that it can accurately tell apart an input signal that has an integer multiple of 1kHz (bin frequencies). In order to increase the resolution of the DFT, one method would be to sample more points, for example sampling 2048 points would give us a resolution of 500Hz. But since we don't have that option as the number of points is already fixed to N=1024, we are left with the option of scaling clock via the MCLK input pin. However, scaling of the MCLK needs to be at a suitable value for the frequency that is to be analysed. Since the measured signal is not always an integer number of periods, this gives rise to a waveform that appears to be distorted as compared to the original sine wave. Such waveforms would have endpoints that are discontinuous and these artificial discontinuities show up in the DFT as high-frequency components not present in the original signal [23]. The energy at one frequency would leak into another, making it difficult to determine the true amplitude of individual peaks. Hence the obtained spectrum would not reflect the actual spectrum, and this is known as spectral leakage. The effect of spectral leakage is minimized by using the windowing technique. The AD5933 uses Hanning (Hann) window which has good side-lobe rejection [15]. Figure 51 shows an illustration of Hann windowing followed by figure 52 that shows the difference between a signal that has no leakage, with leakage and a windowed signal where the leakage is reduced.

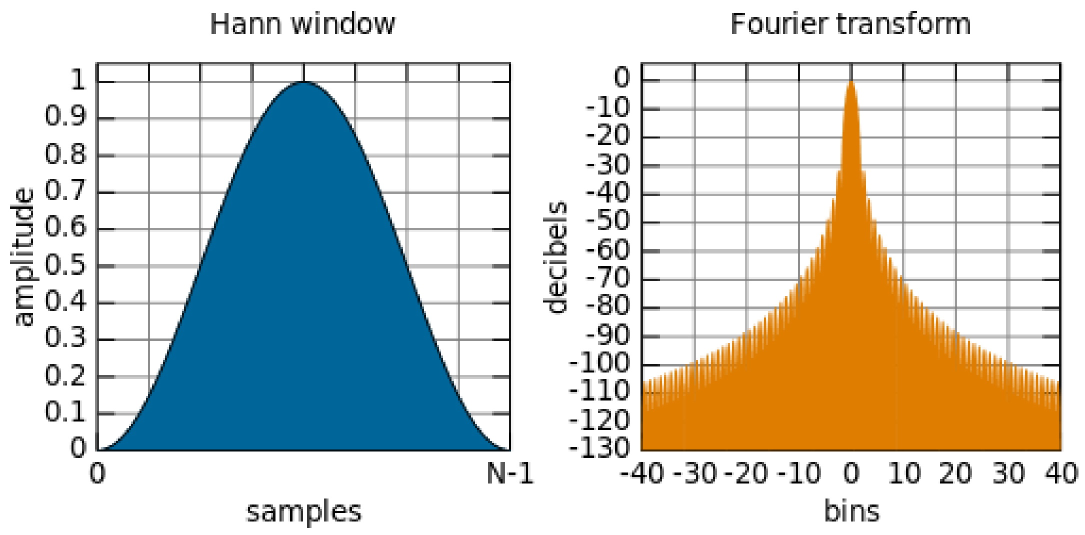


Figure 51: Hann windowing technique to reduce spectral leakage due to input signal being non-integer multiple of fundamental frequency (illustration image is from Wikipedia).

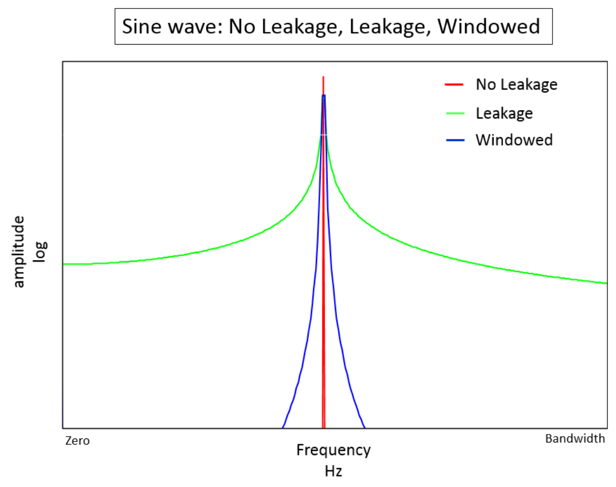


Figure 52: Illustration of a periodic signal that has no leakage, with leakage and a windowed signal where the leakage is reduced. [24]

This figure shows us that by applying windowing, the leakage is now confined over a smaller frequency range, instead of affecting the entire frequency bandwidth of the measurement. By scaling the sampling frequency of ADC at low frequency measurements, we are able to increase the span of the sample window, creating coherent sampling required for accurate results.

To analyse impedance at 9kHz, 5kHz and 3kHz frequencies, the initial MCLK frequency assigned for this range was 1MHz. However, this produced an error as described in the previous section. This is probably due to the fact that at 1MHz of MCLK frequency, the DFT resolution is around 61.035Hz of frequency, causing the samples in each window to be too close to each other, whereby its spectra due to leakage would probably overlap each other. Hence the solution to this is to increase the distance between the samples in each window by decreasing the resolution and increasing the MCLK frequency. In short, the chosen MCLK frequency should not be too low in the case of 9kHz, 5kHz, 3kHz range such that the DFT is forced to resolve at very low difference in frequency. It should not be too high that the span of each window covers very few samples such as in the case of low frequencies, 10Hz-100Hz range.

2) Another issue observed with respect accuracy and clock frequency is that in order to obtain accurate measurements at lower frequencies, the I2C clock frequency had to be reduced to a frequency that is lower than the MCLK frequency. For example during the analysis of measurement at 100Hz -500Hz, the I2C clock produced at the SCL pin of the MCU had to be reduced to 70kHz which is lower than the MCLK frequency of this range at 79kHz. The SCL clock frequency can be changed by writing to the TWBR (TWI Bit

Rate Register) register in the microcontroller. The following formula helps to calculate the right value to write to the TWBR register:

TWBR=((MCU_CLK/SCL_CLK)-16)/2;

Where MCU_CLK refers to the microcontroller system clock, SCL_CLK refers to the I2C clock frequency.

Since the ADC's sampling rate is 16 times lower than the MCLK frequency, having a high I2C clock frequency might mean that the sampling and conversion of signals at 10-100Hz, 100-500Hz range would not have enough time to complete before the results are read in from the AD5933. This issue is not faced at the other frequency ranges because the typical I2C clock frequency is at 100kHz which is very much lower than the MCLK frequencies at those ranges.

In short, the combination of improving on both these factors has helped to improve the accuracy giving us the results in the final results (section 6.2).

3) Once we have an improved accuracy measurement acquisition, the scale at the y-axis becomes more refined and we are able to observe that the signal looks slightly noisy. Ideally for a pure resistor, the magnitude and phase plot should be a smooth, flat line, but in this system there a slight fluctuations around the measured values. This is probably due to the biasing of the front end amplifiers which use the 3.3V power supply signal to re-bias the DC offset. Since the power supply source throughout these measurements is from laptop's USB port which is converted to 3.3V by the Arduino board that was used for its UART module, this signal could be very noisy. However, since the effects on the measurement results were minimal and coupled with time constraint, no improvements

were implemented with regards to this. However, this could be well improved through the use of decoupling capacitors from power supply 3.3V input signal to ground.

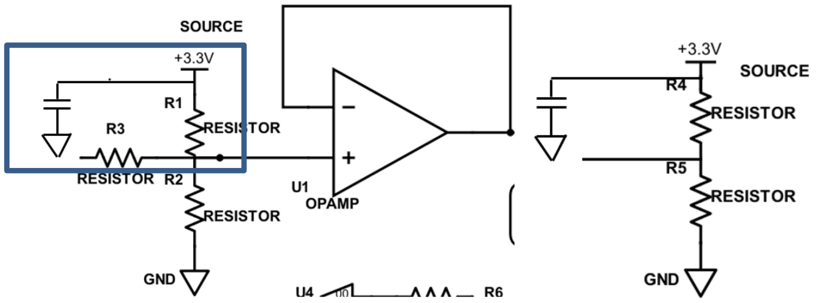


Figure 53: Proposed improvement using decoupling capacitor from power supply input to ground, transmit side (left), TIA biasing (right).

Additionally, decoupling the 3.3V supply to voltage divider biasing network at the TIA using capacitor to ground would be helpful in reducing noise as well. By doing so, any ac signals (power supply hum) is reduced from feeding into the op amp via the power supply line.

CHAPTER 7

CONCLUSION

This chapter summarizes the findings of the project and gives suggestions on possible future work.

A complex impedance measurement system was designed and implemented in this project. The system consists of mainly hardware based circuitry that includes analogue, digital and mixed-signal designs. And all of these were implemented on a 9.85 cm x 5.32 cm sized PCB. MATLAB software was used to carry out the processing of the hardware results. The testing of all components on the PCB were completed. These components which form the circuit blocks of the final design work well as evidenced by the good measurement results. This system managed to implement an effective front-end amplifier topology that is able to convert a 200mV input signal to 10mV impedance excitation voltage without significant noise or signal distortion. The chosen setup is also able to produce the current needed to analyze impedances as low as 10 ohms.

Microcontroller programming codes were developed to perform frequency sweep measurements from 10Hz to 100kHz. The codes work well in delivering the desired performance from the hardware. The sources of error encountered in the initial design were identified and appropriate measures were taken to overcome them to achieve the final design's accuracy. One of the key design choices that influence measurement accuracy is the impedance analysing circuitry's clock frequency when scaling the clock to enable impedance analysis across a wide range of frequency. The clock scaling frequencies need to be chosen after considering the way the impedance analyzer ADC's sampling rate affects the DFT resolution. By considering windowing technique applied in the impedance analyzer's DFT architecture to curb spectral leakage caused by non-integer samples, it is necessary to choose a setting that ensures the span of the of the window is not too big (too few samples) which leads to poor resolution. The span of the window if too small (too many samples) leads to overlapping, distorted spectra which also translates to poor accuracy. Furthermore, to obtain good impedance results at low frequencies, findings show that it is best to keep the I2C clock lower than the input scaling clock so that there is sufficient time for sampling and conversion of the intended signals instead taking measurements at inappropriate timing.

To recap, a system that performs electrochemical impedance spectroscopy of electrochemical cells was built. This system is also suitable for general complex impedance measurement if it is within the impedance range of 10Ω to 100kΩ. Measurements of complex impedances can be carried out without any additional circuitry or having to manually configure any settings as the system can detect the range of impedance magnitude and assign the right setup for accurate measurements. This is possible with the peak detector, MUX and ADC based impedance range detection circuitry that is employed in the system. The system has achieved the specifications intended as shown in the table below.

Power Supply	3.3V
Excitation Frequency:	10Hz to 100kHz
Excitation voltage amplitude:	200mV (board 1) or 10mV (board 2)
Measurable Impedance:	10ohm to 100kohms
Power consumption (active state):	85.4mW

Table 11 : Specifications achieved

At a power consumption of 85.4mW, it is relatively low power when compared to most of other PCB based impedance measurement systems. The measurement results obtained have good accuracy. This is seen in the Bode and Nyquist plots of the measurements plotted against theoretically calculated values. Future work with regards to this system would be to test it on electrochemical sensors. It would be interesting to know if the design works well with the sensors as well as it does with passive R-C components. Parameter extraction of electrochemical cell models using the impedance measurement provided by a system like this can give some insights on the behavior of electrochemical cells, thereby being used to evaluate its performance over time.

REFERENCES

[1] 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," in IEEE Sensors Letters, vol. 2, no. 1, pp. 1-4, March 2018, Art no. 1500104.

[2]Basics of EIS: Electrochemical Research-Impedance. Available at: https://www.gamry.com/application-notes/EIS/basics-of-electrochemical-impedance-spectroscopy/. (Accessed: 2nd Sep 2018)

[3] Electrochemical Impedance and Noise, R. Cottis and S. Turgoose, NACE International, 1999. ISBN 1-57590-093-9.

[4] A. J. Bard and L. R. Faulkner, Electrochemical Methods: Fundamentals and Applications. John Wiley & Sons, Inc, Dec. 2001. [Online]. Available: http://www.wiley.com/WileyCDA/WileyTitle/productCd-0471043729.html

[5] Devangsingh Sankhala, Sriram Muthukumar and Shalini Prasad, "A Four-Channel Electrical Impedance Spectroscopy Module for Cortisol Biosensing in Sweat-Based Wearable Applications," SLAS Technology journal, 2018, pp.1-11

[6] S. Ghoreishizadeh, E. G. Kilinc, C. Baj-Rossi, C. Dehollain, S. Carrara and G. De Micheli, "An implantable bio-micro-system for drug monitoring," 2013 IEEE Biomedical Circuits and Systems Conference (BioCAS), Rotterdam, 2013, pp. 218-221.

[7] 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, Grenoble, 2011, pp. 1-6.

[8] A. Manickam, A. Chevalier, M. McDermott, A. D. Ellington and A. Hassibi, "A CMOS electrochemical impedance spectroscopy biosensor array for label-free biomolecular detection," 2010 IEEE International Solid-State Circuits Conference - (ISSCC), San Francisco, CA, 2010, pp. 130-131.

[9] L. Yang and T. Chen, "A compact signal generation and acquisition circuit for electrochemical impedance spectroscopy," 2016 IEEE Biomedical Circuits and Systems Conference (BioCAS), Shanghai, 2016, pp. 260-263

[10] J. Punter-Villagrasa et al., "A portable point-of-use EIS device for in-vivo biomédical applications," Design of Circuits and Integrated Systems, Madrid, 2014, pp. 1-6.

[11] Diming Zhang, Yanli Lu, Qian Zhang, Lei Liu, Shuang Li, Yao Yao, Jing Jiang, Gang Logan Liu, Qingjun Liu, “Protein detecting with smartphone-controlled electrochemical impedance spectroscopy for point-of-care applications,’’ Sensors and Actuators B: Chemical, Volume 222, 2016, pp. 994-1002.

[12] P. Bogonez-Franco, L. Nescolarde, C. Galvez-Monton , R. Bragos, and J. Rosell-Ferrer, “An implantable bioimpedance monitor using 2.45 GHz band for telemetry,” Physiol. Meas., 34, 2013

[13] Evaluating the AD5933 1 MSPS, 12-Bit Impedance Converter Network Analyzer, User Guide 364, Analog Devices

[14] Analog.com. (2018). AD5933 Datasheet. [online] Available at: http://www.analog.com/media/en/technical-documentation/data-sheets/AD5933.pdf [Accessed 26 Aug. 2018].

[15] Sean Brennan, Measuring a Loudspeaker Impedance Profile Using the AD5933, Application Note 843, Analog Devices

[16] High Accuracy Impedance Measurements Using 12-Bit AD5933 Impedance Converters, Circuit Note 217, Analog Devices

[17] Ducu, D. Op Amp Rectifiers, Peak Detectors and Clamps; Technical Report DS01353A; Microchip Technology Inc.: Chandler, AZ, USA, 2011. [Google Scholar]

[18] Analog.com. (2018). AD8606 Datasheet. [online] Available at: http://www.analog.com/media/en/technical-documentation/data-sheets/AD8605_8606_8608.pdf [Accessed 26 Aug. 2018].

[19] Analog.com. (2018). LTC6268 Datasheet. [online] Available at: http://www.analog.com/media/en/technical-documentation/data-sheets/62689f.pdf [Accessed 26 Aug. 2018].

[20] Maximintegrated.com (2018). DS1077 Datasheet. [online] Available at: https://www.maximintegrated.com/en/products/digital/clock-generation-distribution/silicon-crystal-oscillators/DS1077.html [Accessed 26 Aug. 2018].

[21]Analog.com. (2018). ADG804 Datasheet. [online] Available at: http://www.analog.com/media/en/technical-documentation/data-sheets/ADG804.pdf [Accessed 26 Aug. 2018].

[22] Microchip.com (2018). Atmega-328p Datasheet. [online] Available at: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-42735-8-bit-AVR-Microcontroller-ATmega328-328P_Datasheet.pdf [Accessed 26 Aug. 2018].

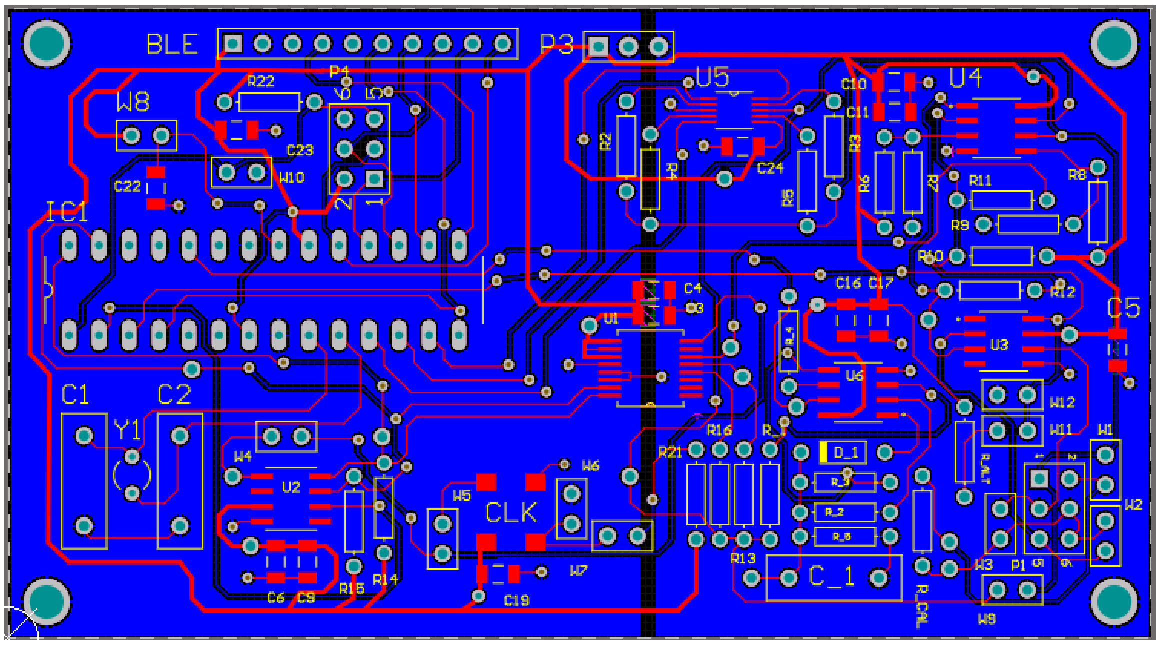
[23] National Instruments. Understanding FFTs and Windowing [Online]. Available: http://download.ni.com/evaluation/pxi/Understanding%20FFTs%20and%20Windowing.pdf (Accessed: 2nd Sep 2018)

[24] Siemens. Windows and Spectral Leakage [Online]. Available: https://community.plm.automation.siemens.com/t5/Testing-Knowledge-Base/Windows-and-Spectral-Leakage/ta-p/432760 (Accessed: 2nd Sep 2018)

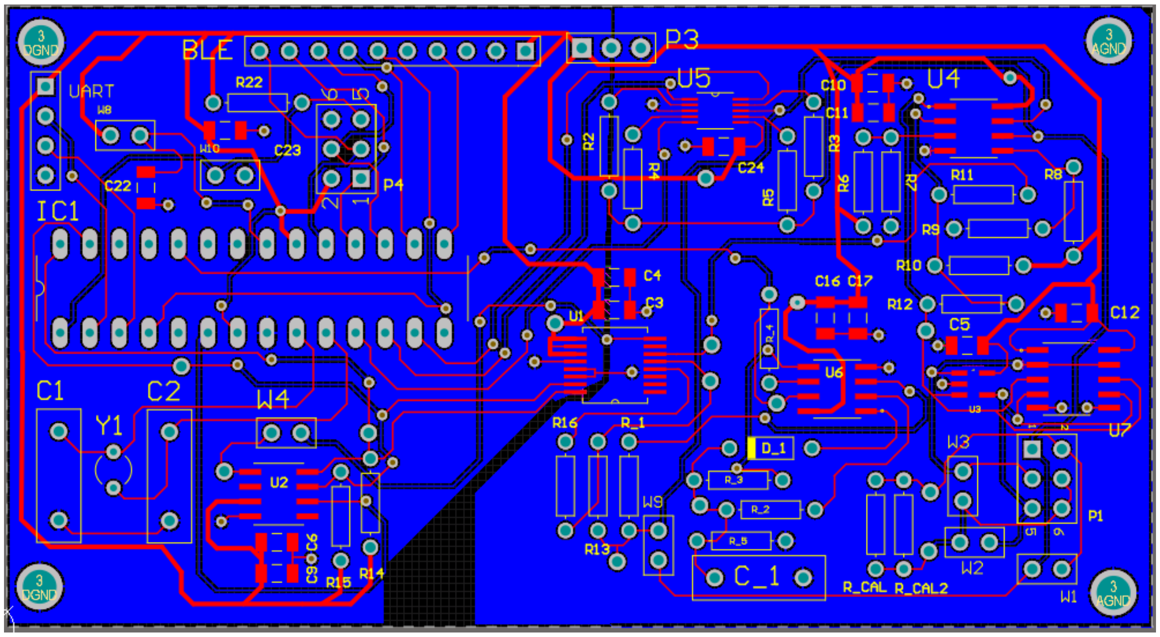
APPENDIX

PCB LAYOUTS

PCB Layout of board 1

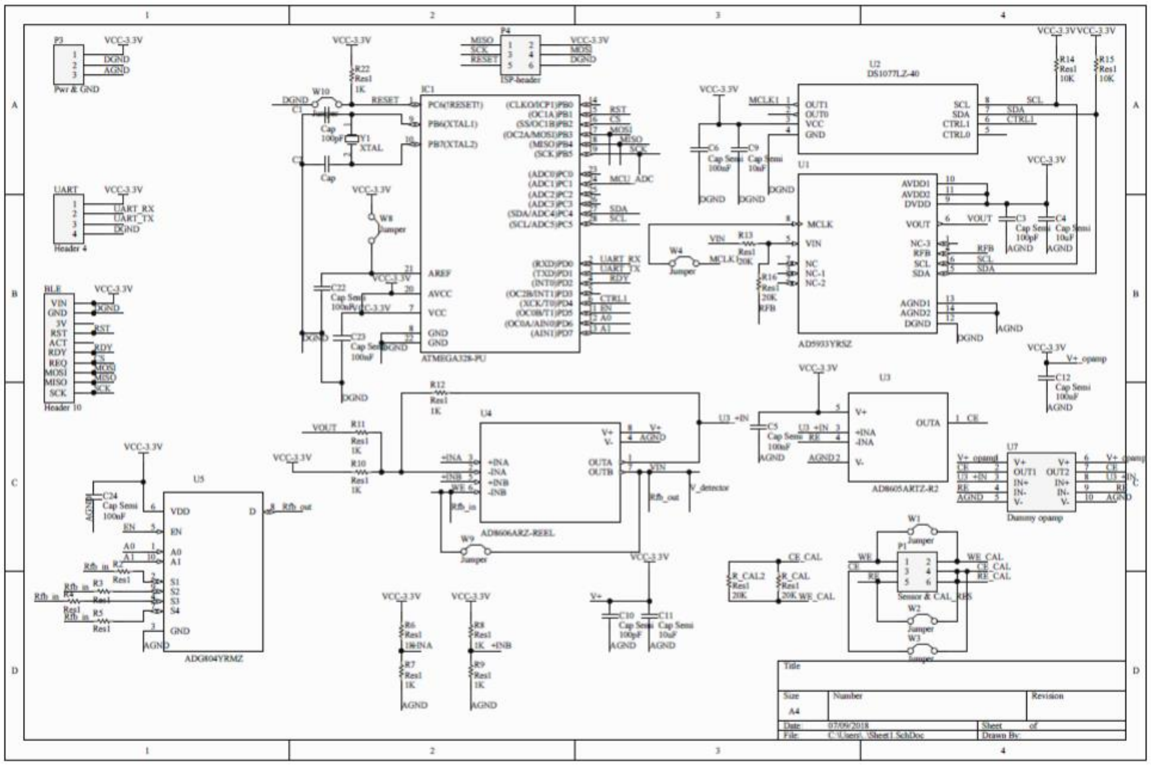


PCB Layout of board 2

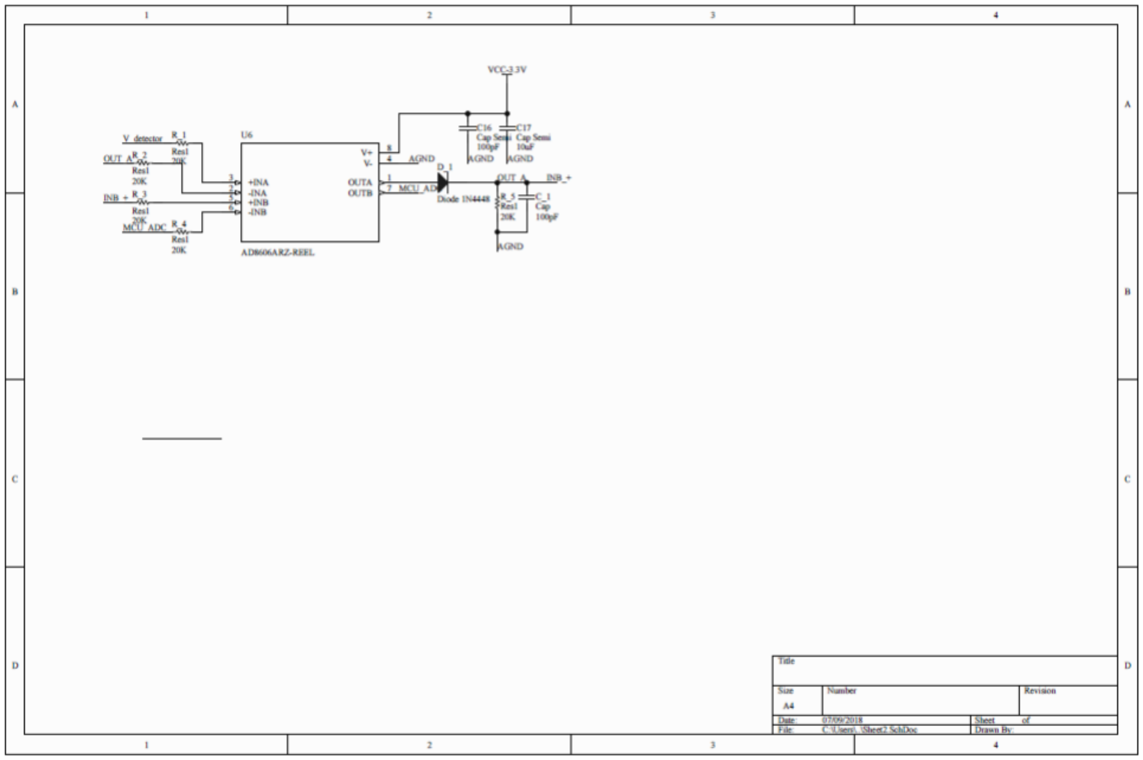


Final Design Schematics

Page 1



Page 2



Hardware codes - Main program

```
/*
 * developed_codes_v2.c
 *
 * Created: 12/08/2018 21:24:01
 * Author : Vinnothini.R
 */

//EIS main program

#include <stdio.h>
#include <avr/io.h>
#include <math.h>
#include <util/delay.h>
#include "frequency_values.h"
#include "i2c.c"
#include "DS1077.c"
#include "myUSART.c"
#include "myADC.c"

#define F_CPU 8000000UL

//AD5933 Register addresses
#define Control_high 0x80
#define Control_low 0x81
#define Freq_high 0x82
#define Freq_mid 0x83
#define Freq_low 0x84
#define FreqInc_high 0x85
#define FreqInc_mid 0x86
#define FreqInc_low 0x87
#define NumInc_high 0x88
#define NumInc_low 0x89
#define NumSettle_high 0x8a
#define NumSettle_low 0x8b
#define Status 0x8f
#define Real_high 0x94
#define Real_low 0x95
#define Imag_high 0x96
#define Imag_low 0x97
#define incsteps_reg 0x88

int16_t Data_proc(unsigned char data_high, unsigned char data_low)
{
       int16_t data;

       data=(int16_t)data_high*256+data_low;
       // this converts from 2's complement to binary where if data<= 0x7fff then its a
positive number,  aka the 16th bit is 0
       if(data > 0x7fff)
       {
              data=0x10000-data;  //i have proved this using online compiler & hand
calculation method of converting 2's complement to normal binary
       }
       return data;
}


void main(void)
```

```
{
       unsigned char a,b,c,d;
       int16_t R,I;
       int i,j, freq;              //Real and Imag numbers
       uint8_t f_arr1[3], f_arr2[3], prev_range=0, res_range=0;  //char -> 8 bits
       uint8_t flag=0;
       uint16_t cnt, freq_value=0; // long

       char s[20];
       char r[20];
       char x[20];
       char y[20];

       //initialize ports for RFB_MUX PORTD PD5, PD6, PD7
       DDRD |= (1<<DDD0)|(1<<DDD5)|(1<<DDD6)|(1<<DDD7); //ports as output

       PORTD|= (1<<PORTD5); //make D5 output high
       PORTD&=~(1<<PORTD6)|(1<<PORTD7);
         //make D6, D7 output low
       USART_init();
       Init_TWI();
       init_DS_1077();

       Byte_write(Control_low, 0x10);
       Byte_write(Control_low, 0x08);// for ext clock -mclk,  its 0x08
       Byte_write(Control_high, 0xb0);

       for (i=0; i<11;  i++){
              if (i==11){
                     Init_TWI_slowclk2(); }
                     cnt=0;  // initialize cnt that stores evaluating freq value
                     if(i>1)
                     {
                     prev_range= freq_range[i-1];
                     }

                     if (freq_range[i]!=1 && freq_range[i]!= prev_range)
                     {
                     //then call DS_1077 function to change MCLK freq
                     MCLK_config(freq_range[i]);
                     }

       f_arr1[0]=start_freq[i][j];  f_arr1[1]=start_freq[i][j+1];
       f_arr1[2]=start_freq[i][j+2];

       Block_write(Freq_high, 3, f_arr1);  //am writing my start frequency here

       f_arr2[0]=inc_freq[i][j];  f_arr2[1]=inc_freq[i][j+1];
       f_arr2[2]=inc_freq[i][j+2];
       Block_write(FreqInc_high, 3, f_arr2);//am writing my frequency increment here

       Byte_write(NumInc_high, 0x00);  //maximum number of increments is 511
       Byte_write(NumInc_low, 0x8f);  //LSB: D7-D0
              //Set Num of Settling Time Cycles
       Byte_write(NumSettle_high, 0x00);
       Byte_write(NumSettle_low, 0x0f);  //MSB: D7-D0 : this is 15 cycles
       //initialize
       Byte_write(Control_high, 0x13);
       _delay_ms(5000);
              if(flag==0){
                     res_range=peak_detect();
                     flag=1;
                     itoa(res_range, x, 10);
                            USART_CharTransmit(x);
                            USART_CharTransmit(",");
                     }

                     if(res_range==4){
                     //start sweep
                     Byte_write(Control_high, 0x22);
                     }
                     else{
                            Byte_write(Control_high, 0x23);
                     }



                     cont:
                     while(!(Byte_read(Status) & 0x02));

                     a=Byte_read(Real_high);//real register values
                     b=Byte_read(Real_low);
                     R=Data_proc(a, b);
                     itoa(R, s, 10);
                     USART_CharTransmit(s);

                     USART_CharTransmit(",");

                     c=Byte_read(Imag_high);  //imag register values

                     d=Byte_read(Imag_low);

                     I=Data_proc(c, d);
                     itoa(I, r, 10);
                     USART_CharTransmit(r);
                     USART_CharTransmit(",");

                     freq_value=start_freq_deci[i]+cnt;
                     itoa(freq_value, s, 10);
                     USART_CharTransmit(s);
                    USART_CharTransmit(",");
                     cnt=cnt+inc_freq_deci[i];


                     if((Byte_read(Status) & 0x04)==0)
                     {
                            _delay_ms(3000);
                            if(res_range==4){
                            Byte_write(Control_high, 0x32);
                            }
                            else{
                                   Byte_write(Control_high, 0x33);
                            }
                            goto cont;
                     }
                     else
                     {
                            Byte_write(Control_high, 0xb0);  //standby

                     }
              }
}
```

DS1077.c

```
//DS master clock is 40MHz and can be divided down to 4.8kHz
#include <avr/io.h>
#include <util/delay.h>
#include "i2c.h"
#define MT_DATA_ACK 0x18

//DS1077-40 address
#define SLA_W_DS 0Xb0
#define SLA_R_DS 0Xb1

//DS1077-40 command byte
#define ACCSS_DIV 0x01
#define ACCSS_MUX 0x02
#define ACCSS_BUS 0x0D
#define WRT_E2 0x3F

#define MUX_PDN1 0  // PDN1=0, functions as enable for OUT1 only
#define MUX_PDN0 0  // CTRL0 is det by EN0,SEL0 values NOTE:  don't set to 1 as it will
cause CTRL0 to perform power down rgsless of othr bits!
#define MUX_SEL0 1  // output will be master clock,  if =1 out freq of the M prescaler
(check this when testing!)
#define MUX_EN0 0   // CTRL0 is det by SEL0, PDN0
#define MUX_DIV1 0  // if =1,  N divider is ignored,  and P1 value is routed directly to
OUT1 pin
#define MUX_0M1 0   //  these bits mean that prescalers P0 is 1 (default)
#define MUX_0M0 0

void two_byte_write(uint8_t cmd, uint8_t msb,  uint8_t lsb){
       Init_TWI();
       Send_start();
       Send_byte(SLA_W_DS);
       Send_byte(cmd);
       Send_byte(msb);
       Send_byte(lsb);
       Send_stop();
}


uint8_t get_mux_msb(uint8_t MUX_1M1){
uint8_t val = 0x00;
       val = val
       | (MUX_PDN1 << 6)
       | (MUX_PDN0 << 5)
       | (MUX_SEL0 << 4)
       | (MUX_EN0 << 3)
       | (MUX_0M1 << 2)
       | (MUX_0M0 << 1)
       | (MUX_1M1);
       return val;
}
uint8_t get_mux_lsb(uint8_t MUX_1M0){
       uint8_t val = 0x00;
       val = val
       | (MUX_1M0 << 7)
       | (MUX_DIV1 << 6);
       return val;
}
```

```
//initial clock controlling for range 1
void init_DS_1077(void){
       uint8_t msb_mux, lsb_mux,  DIV_MSB, DIV_LSB,  MUX_1M1,  MUX_1M0;
       DIV_MSB= 0x00;  //N9 to N2
       DIV_LSB= 0xc0;  //N1 and N0
       //P1 is 00-1,  01-2,  10-4,  11-8
       MUX_1M0=0;  //initially i want 8MHz so prescaler should be 1
       MUX_1M1=0;
       msb_mux = get_mux_msb(MUX_1M1);
       lsb_mux = get_mux_lsb(MUX_1M0);
       two_byte_write(ACCSS_MUX, msb_mux, lsb_mux);
       _delay_ms(5000);
       two_byte_write(ACCSS_DIV, DIV_MSB, DIV_LSB);
       return(0);
}

//subsequent clock controlling for range 2-5
void MCLK_config(uint8_t range)
{
       uint8_t msb_mux, lsb_mux,  DIV_MSB, DIV_LSB,  MUX_1M1,  MUX_1M0;

 MUX_1M0=1;
 MUX_1M1=1;

       switch(range){
              case 2:
               DIV_MSB= 0x00;  //N9 to N2
               DIV_LSB= 0x00;  //N1 and N0   Div=5
                     msb_mux = get_mux_msb(MUX_1M1);
                     lsb_mux = get_mux_lsb(MUX_1M0);
               two_byte_write(ACCSS_MUX, msb_mux, lsb_mux);  //make prescaler =8
starting from range 2 onwards
               _delay_ms (5000);
               two_byte_write(ACCSS_DIV, DIV_MSB, DIV_LSB);
               break;

              case 3:
               DIV_MSB= 0x10;  //N9 to N2
               DIV_LSB= 0x80;  //N1 and N0   Div=6

               two_byte_write(ACCSS_DIV, DIV_MSB, DIV_LSB);
               break;

              case 4:
               DIV_MSB= 0x0f;  //N9 to N2
               DIV_LSB= 0x40;  //N1 and N0   Div=63 1111 0100
               two_byte_write(ACCSS_DIV, DIV_MSB, DIV_LSB);
                break;

              case 5:

              DIV_MSB= 0x67;  //N9 to N2
              DIV_LSB= 0xc0;  //N1 and N0   Div=417
              two_byte_write(ACCSS_DIV, DIV_MSB, DIV_LSB);
              break;
              }

       return(0);
}
```

i2c.c

```
/*
 *i2c.c
 *
 * Created: 02/07/2018 18:47:54
 * Author : Vinnothini.R
 */

#include <avr/io.h>
#include <stdio.h>
// AD5933
#define ACK 0xc4
#define SUCCESS 0xff //Flag
#define SLA_W 0x1a  //slave address - 0001101 for AD5933 followed by 0 bit for write
#define SLA_R 0x1b  //slave address - 0001101 for AD5933 followed by 1 bit for read
#define MT_SLA_ACK 0x18
#define MR_SLA_ACK 0x40
#define MT_DATA_ACK 0x28
#define F_CPU 8000000UL

//unsigned char Init_TWI(void)
int Init_TWI(void)
{
       TWBR = 0x20; // 100kHz
       TWCR = 0x04;         //Enable TWI-interface (set TWEN to 1)
       return 1;
}
int Init_TWI_slowclk(void)
{
       TWBR = 0x32; //try 70kHz
       TWCR = 0x04;         //Enable TWI-interface (set TWEN to 1)
       return 1;
}
int Init_TWI_slowclk2(void)
{
       TWBR = 0x164; //try 11kHz
       TWCR = 0x04;         //Enable TWI-interface (set TWEN to 1)
       return 1;
}

void Wait_TWI_int(void)
{
       while(!(TWCR & (1<<TWINT))); // syntax from datasheet
}
unsigned char Send_start(void)
{
       TWCR = (1<<TWINT)|(1<<TWSTA)|(1<<TWEN)    ;      //Send START

       Wait_TWI_int();             //Wait for TWI interrupt flag to be set

       if((TWSR & 0xF8)!=0x08 || (TWSR & 0xF8)!=0x10)  //
       return TWSR;  }
void Send_stop(void)
{
       TWCR = (1<<TWINT)|(1<<TWEN)|(1<<TWSTO);         //Send a STOP condition
       while(TWCR & (1<<TWSTO));
}
```

```
unsigned char Send_adr(unsigned char adr)
{
       Wait_TWI_int();             //Wait for TWI interrupt flag set
       TWDR = adr;
      TWCR = (1<<TWINT) |(1<<TWEN);  // cLEAR INT flag to start transmission
       Wait_TWI_int();             //Wait for TWI interrupt flag set

       if((TWSR & 0xF8)!= MT_SLA_ACK || (TWSR & 0xF8)!= MR_SLA_ACK ||(TWSR &
0xF8)!=MT_DATA_ACK){
       return TWSR;  //If NACK received return TWSR
       }
}
unsigned char Send_byte(unsigned char data)
{
       Wait_TWI_int();
       TWDR = data;
       TWCR = (1<<TWINT) |(1<<TWEN);  // cLEAR INT flag to send byte

       Wait_TWI_int();             //Wait for TWI interrupt flag set

       if((TWSR & 0xF8)!= MT_DATA_ACK){
       return TWSR;  //If NACK received return TWSR
       }
}

unsigned char Set_pointer(unsigned char reg_loc)
{
       Send_start();
       Send_adr(SLA_W);
       Send_byte(0xb0);     //Pointer command code '1011 0000' in AD5933
       Send_byte(reg_loc);  //a register location at which the pointer points
       return 1;
}

unsigned char Byte_write(unsigned char reg_addr, unsigned char data)
//Write a byte to AD5933.
{
       Send_start();
       Send_adr(SLA_W);
       Send_byte(reg_addr);
       Send_byte(data);
       Send_stop();
       return 1;
}

unsigned char Block_write(unsigned char reg_loc, unsigned char byte_num, uint8_t
data_p[3])  //Write a block of data to AD5933.
{
       unsigned char i;
       char y[20];
       Set_pointer(reg_loc);       //set the pointer location
       Send_start();       //write the data block
       Send_adr(SLA_W);
       Send_byte(0xa0);     //Block write command code '1010 0000'
       Send_byte(byte_num); //Num of data to be sent

       for(i = 0;i < byte_num;i++)  //Send the data bytes
       {
              Send_byte(data_p[i]);
       }
       Send_stop();
       return 1;
}

unsigned char Byte_read(unsigned char reg_loc) //Read a byte from AD5933.
{
	Set_pointer(reg_loc);		//set the pointer location

	//Receive a byte
	Send_start();
	Send_adr(SLA_R);
	TWCR = (1<<TWINT) |(1<<TWEN); // clear int flag
	Wait_TWI_int();		//Wait for TWI interrupt flag set
	return TWDR;
}
```

frequency_values.h

```
//frequency_values.h
//--------------------------------------90k-100kHz------------         -------10k-
10.9k---------30k-30.2k------
static const uint8_t start_freq[12][3]={{0x5c, 0x25, 0xd8}, {0x47, 0xab, 0xa8}, {0x0a,
0x3d, 0x18}, {0x1e,0xb7, 0x48},{0x09,0x36,0xfc}, {0x10,0x62,0x4e},{0x09, 0xd4, 0x94},\
{0x08,0xd9,0x03}, {0x06, 0xe1, 0xad}, {0x04, 0xea, 0x57},{0x0a, 0x52, 0x6c},\
{0x06,0xd5,0x10}};
static const uint8_t inc_freq[12][3]={{0x00, 0x1a, 0x36}, {0x00, 0x1a, 0x36},\
{0x00,0x02,0x9f},  {0x00, 0x00, 0x86},{0x00,0x02,0x9f}, {0x00,0x08,0x64},\
{0x00, 0x04, 0x32},{0x00, 0x19, 0x2b}, {0x00, 0x05,0x08}, {0x00, 0x0c, 0x95},\
{0x00, 0x84, 0x1f}, {0x00, 0xae, 0xe8}};
static const uint16_t start_freq_deci[12]={9000, 7000, 10000, 30000, 9000, 5000, 3000,
900, 700, 500, 100, 10};//actual frequency divided by 100 for 1st, 2nd frequency.
static const uint16_t inc_freq_deci[12]={100, 100, 10, 2, 10, 10, 5, 10, 2, 5, 5, 1};
static const uint8_t freq_range[12]={1, 1, 1, 1, 1, 2, 2, 3, 3, 3, 4, 5};
```

myUSART.c

```
#define F_CPU 8000000UL
#include <avr/io.h>
#include <util/delay.h>

#define BAUDRATE 9600
#define BAUD_PRESCALLER (((F_CPU / (BAUDRATE * 16UL))) - 1)

void USART_init(void){

	UBRR0H = (uint8_t)(BAUD_PRESCALLER>>8);
	UBRR0L = (uint8_t)(BAUD_PRESCALLER);
	UCSR0B |= (1<<RXEN0)|(1<<TXEN0);
	//UCSR0C = (3<<UCSZ00);
	UCSR0C |= (1<<USBS0)|(3<<UCSZ00);
}

unsigned char USART_receive(void){

	while(!(UCSR0A & (1<<RXC0)));
	return UDR0;

}

void USART_send( unsigned char data){

	while(!(UCSR0A & (1<<UDRE0)));
	UDR0 = data;

}

void USART_CharTransmit(char* data)
{
	int n;
	n=0;
	while (1)
	{
		while ((UCSR0A & (1 << UDRE0)) == 0) {};
		UDR0 =*(data+n);
		n++;
		if(!(*(data+(n)))){
			break;
		}
	}
}
```

myADC.c

```
#include <avr/io.h>
#include <math.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include "myUSART.h"
#define F_CPU 8000000UL
void ADC_init(void)
{
       PRR &=~(1<<PRADC);
       ADMUX|=(1<<REFS0)|(1<<MUX0);
       ADCSRA|=(1<<ADEN)|(1<<ADPS0)|(1<<ADPS2); //no interrupt
};

uint16_t ReadADC()
{

       ADMUX|=(1<<MUX0); //  my case MUX_ADC is connected to ADC1 so 0001
       //Start Single conversion
       ADCSRA|=(1<<ADSC);

       //Wait for conversion to complete means ADIF becomes set to 1
       while(!(ADCSRA & (1<<ADIF)));

       //Clear ADIF by writing one to it (based on datasheet)
       ADCSRA|=(1<<ADIF);

       return(ADC);
}
uint16_t Average_readADC(){
       uint8_t i;
       uint16_t sum=0,val=0,average_adc=0;
       for(i=0;i<10;i++){
              val=ReadADC();
              sum=sum+val;
       }
       average_adc=sum/10;
       return average_adc;
}

uint8_t peak_detect(void)
{
       uint16_t adc_result=0;
       uint8_t res_range=0, count=0;
       uint16_t lower_lim=541;
       uint16_t upper_lim=848;
       char s[20];

       ADC_init();

 check:    adc_result=ReadADC();  // Read Analog value from channel 1

           if (adc_result>=lower_lim && adc_result<=upper_lim){
                     res_range=res_range+1; //A0:D6=0  A1:D7=0
                            }else{
                            count=count+1;
                            change_muxout(count);
                            res_range=count;
                            if(count<4){
                              goto check;
                            }
                      }
return res_range;
}


void change_muxout (uint8_t count){

       switch(count){
              case 1:
        PORTD|=(1<<PORTD6);//A0:D6=1  A1:D7=0
        PORTD&=~(1<<PORTD7);//change mux output

              break;

              case 2:
              PORTD|=(1<<PORTD7);//A0:D6=0  A1:D7=1
              PORTD&=~(1<<PORTD6);//change mux output

              break;

              case 3:
              PORTD|=(1<<PORTD6);//A0:D6=1  A1:D7=1
              PORTD|=(1<<PORTD7);//change mux output

              break;

              default:
              ;
              break;
       }
     return (0);
       };
```

MATLAB codes

```
my_data=randles5;
[row,col]=size(my_data);
new_row=(col-1)/3;
range=my_data(1,1);
new_matrix=my_data(2:col);
var=zeros(new_row,3);
Impedance_mag=zeros(new_row,1);
phase_unknown=zeros(new_row,1);
phase=zeros(new_row,1);
theta=zeros(new_row,1);
for x=1:new_row
    var(x,:)=new_matrix(3*x-2:3*x);
end

R=var(:,1);
I=var(:,2);
frequency=var(:,3);
frequency_1 = [10*frequency(1:20);frequency(21:end)]; %represents 90 000 and
70 000 correctly
temp=(R.^2+I.^2);
mag=sqrt(temp);

for y=1:new_row
     if ((R(y,:)>0) && (I(y,:)>0))
         theta(y,:)=atan(I(y,:)/R(y,:));
         phase_unknown(y,:)=(theta(y,:)*180)/pi;
     elseif ((R(y,:)>0) && (I(y,:)<0))
         theta(y,:)=atan(I(y,:)/R(y,:));  %4th quad theta= -ve angle
         phase_unknown(y,:)=(theta(y,:)*180)/pi +360;
     elseif ((R(y,:)<0)&&(I(y,:)<0))
         theta(y,:)=atan(I(y,:)/R(y,:));  %3rd quad theta is +ve
         phase_unknown(y,:)=((theta(y,:)*180)/pi)+180;
     elseif ((R(y,:)<0)&&(I(y,:)>0))
         theta(y,:)=atan(I(y,:)/R(y,:));  %2nd quad is -ve
         phase_unknown(y,:)=((theta(y,:)*180)/pi)+180;
     end
end

             if range(y,:)==4
             Impedance_mag(y,:)=(gain_r4(y,:).*mag(y,:)).^(-1);
phase(y,:)=sys_phase4(y,:)-phase_unknown(y,:);
             elseif range(y,:)==3
             Impedance_mag(y,:)=(gain_r3(y,:).*mag(y,:)).^(-1);
              phase(y,:)=sys_phase3(y,:)-phase_unknown(y,:);
             elseif range(y,:)==2
             Impedance_mag(y,:)=(gain_r2(y,:).*mag(y,:)).^(-1);
               phase(y,:)=sys_phase2(y,:)-phase_unknown(y,:);
             elseif range(y,:)==1
             Impedance_mag(y,:)=(gain_r1(y,:).*mag(y,:)).^(-1);
               phase(y,:)=sys_phase1(y,:)-phase_unknown(y,:);
             end
```

```
for y=1:new_row
        if (phase(y,:)>180)
           phase(y,:)=phase(y,:)-360;
        end
        if (phase(y,:)<-180)
        phase(y,:)=360+phase(y,:);
       end
   end

frequency_2 =
[frequency_1(111:end);frequency_1(101:110);frequency_1(91:100);frequency_1(
81:90);frequency_1(71:80);frequency_1(61:70);frequency_1(51:60);frequency_1
(41:50);frequency_1(21:40);frequency_1(11:20);frequency_1(1:10);];
Impedance_mag_2 =
[Impedance_mag(111:end);Impedance_mag(101:110);Impedance_mag(91:100);Impeda
nce_mag(81:90);Impedance_mag(71:80);Impedance_mag(61:70);Impedance_mag(51:6
0);Impedance_mag(41:50);Impedance_mag(21:40);Impedance_mag(11:20);Impedance
_mag(1:10);];
phase_2=[phase(111:end);phase(101:110);phase(91:100);phase(81:90);phase(71:
80);phase(61:70);phase(51:60);phase(41:50);phase(21:40);phase(11:20);phase(
1:10);];
theta_1=(phase_2*pi)/180;
Real=Impedance_mag_2.*cos(theta_1);
Imag=Impedance_mag_2.*sin(theta_1);

figure(1);
plot(Real,-Imag,'o');
title('Nyquist Plot');
% legend('Theoretical');
hold on

plot(Real_plot,-Imag_plot,'m');
legend('Measured','Theoretical');
xlabel('Z, Real') % x-axis label
ylabel('-Z",Imaginary') % y-axis label

figure(2);
 semilogx(frequency_2,abs(Impedance_mag_2),'o');
 title('Bode Plot');
 hold on
 semilogx(frequency_2,Z_mag,'m');
 legend('Measured','Theoretical');
 xlabel('Frequency (Hz)') % x-axis label
ylabel('Magnitude(Ohms)') % y-axis label
figure(3);
 semilogx(frequency_2,phase_2,'o');
 title('Bode Plot');
 hold on
 semilogx(frequency_2,phase_test_data,'m');
 legend('Measured','Theoretical');
 xlabel('Frequency (Hz)') % x-axis label
ylabel('Phase (degrees)') % y-axis label
```