University of Manitoba
Department of Electrical & Computer Engineering

ECE 4600 Group Design Project

Final Project Report

Design and Implemention of a Frequency Response Analysis System

by
Group 02

Alan Mark
Naima Shahzadi
Hao Liang Deng
Tong Shu

Academic Supervisor(s)
Behzad Kordi, Ph.D, P. Eng

Industry Supervisors
Nathan Jacob, P. Eng – Manitoba Hydro

Date of Submission
March 4, 2015

Abstract

Power transformers are one of the most important elements of a power systems network. Proper maintenance and repairs are key aspects to the longevity of a power transformer. Frequency respone analysis (FRA) systems are used to determine any mechanical or electrical displacements that may occur within the transformer by measuring the transfer function magnitude and phase over a frequency range [1].

The proposed project details the design and implementation of a FRA system. The project goal consisted of determining the transfer function gain, V_{out}/V_{in}, and the phase of a two-port network over a frequency range of 10 Hz to 2 MHz. The design consists of two major components, the interface and coding, and the hardware components. The design consisted of using a variable-frequency oscillator controlled by a microcontroller to produce a sine wave that is then injected into an amplifier. The amplifier will ensure that adequate voltage and power is provided to the test unit. A detector is then used to determine the gain and phase of the test unit and passes these values to the microcontroller. A system interfacing program then plots the data on a PC and on a online database program. The FRA system design produced a fully functioning FRA system and allowed system tests to be completed on a low-pass filter, and a band-pass filter that closely resembles the equivalent circuit model of a power transformer winding.

Contributions

	Alan Mark	Hao Liang Deng	Naima Shahzadi	Tong Shu
Research	○	○	○	○
Microcontroller and VFO	○	●	○	○
Amplifier	○	○	●	○
Gain and Phase Detector	●	○	○	○
System Interface	○	○	○	●
Reports	○	○	○	○

Legend: ● Lead task ○ Contributed

Acknowledgements

This project could not be completed without the help and advice provided by many people. We would like to acknowledge their contributions and support to help accomplish the project.

Thanks to Dr. Behzad Kordi, our supervisor, for the guidance and knowledge provided to us for the entire duration of the project. Without his help, the project would not have achieved the intended goal.

Thanks to Daniel Card, the engineer-in-residence, for all the help provided when we needed consultation.

Thanks to the ECE tech shop, for helping us order parts and providing us with the breadboards, wire-wrap board, circuit components and other necessary equipment to complete the project.

Thanks to Sinisa Janjic, for creating the necessary adapters to allow proper fitting of the hardware components to a breadboard.

Thanks to Aidan Topping and Rafi Mahabbat Bin Belal for providing feedback on our reports.

Table of Contents

List of Figures

List of Tables

Nomenclature

ADC	Analog-to-Digital converter
CSV	Comma separated values
DAC	Digital-to-analog converter
FRA	Frequency response analysis
IDE	Integrated Development Environment
IF	Intermediate frequency
ODAP	Online data analysis program
PCB	Printed circuit board
RF	Radio frequency
SR	Slew rate
VFO	Variable frequency oscillator

Chapter 1

Introduction

1.1 Motivation

Power transformers are one of the most important elements in a power system network, thus accurate diagnostics and proper maintenance of a power transformer are crucial. Power transformers can experience electrical and mechanical displacements, such as winding deformation, during its regular operation or when being transported. These displacements or faults can reduce the life expectancy and lead to more severe problems or even complete failure of the power transformer. Frequency response analysis works by determining the transfer function, V_{out}/V_{in}, and the phase difference between the input and output waveform over a range of frequencies [1]. FRA results produce the fingerprint of a transformer. These fingerprints are compared with the baseline, provided by the manufacturer, or previous ones and are used to determine any displacements and where it has occurred within the transformer, allowing a more effective and efficient repair, and reducing undesired repair costs. [1]

1.2 Project Objective

The goal of this project is to design a frequency response analysis system that will determine the magnitude and phase of the transfer functionof a two-port network over a frequency range. The project will be a cost effective solution to FRA systems because the current market devices are in excess of thousands of dollars whereas the total budget of this project did not exceed $400.00

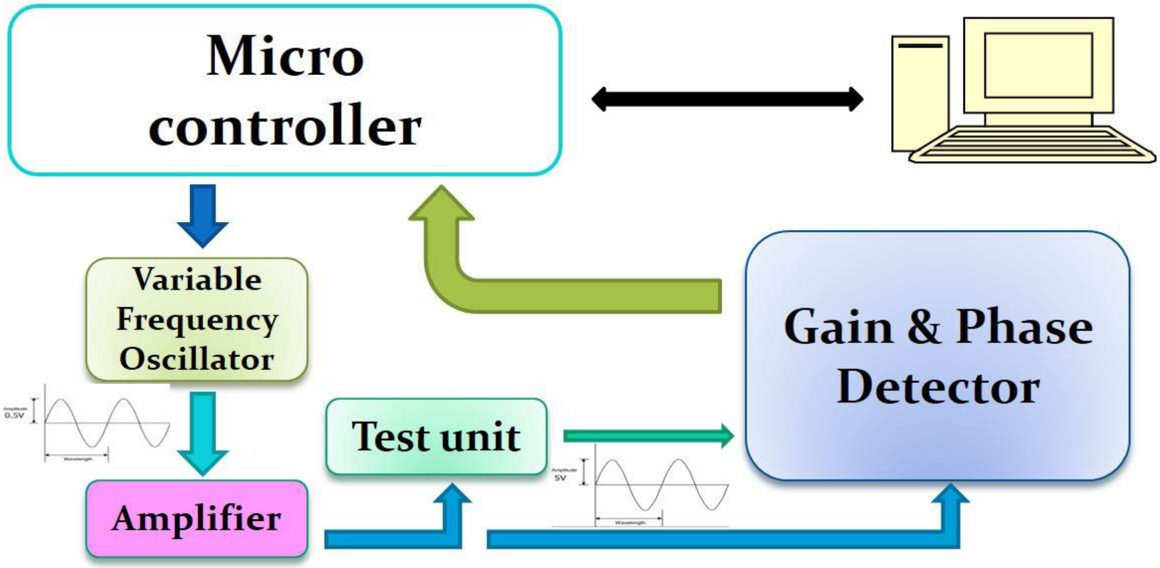


Fig. 1.1: System flow chart

1.3 Project Description

The project will be designed to develop a FRA system by determining the transfer function magnitude V$_{out}$/V$_{in}$ and phase angle over a frequency range of 10 Hz to 2 MHz for a two-port network. The system flow chart can be seen in Fig. 1.1. To achieve this objective, a system interface will communicate with the microcontroller to create a variable frequency sinusoidal waveform. This waveform will then be amplified to ensure adequate power is supplied to the test unit. After amplification, the waveform will be injected into the input of the test unit. After allowing steady state to be reached, the input and output of the test unit will be measured to determine the amplitude of V_{out}/V_{in} and phase difference. These results will then be displayed on the system interface. To achieve accurate results, the system requirements are as follows. The frequency range must be variable from 10 Hz to 2 MHz. The test result, which includes the phase difference and the amplitude of V_{out}/V_{in} over the entire frequency range, will be stored in a comma-separated values (CSV) file. Finally, the system interface will produce a plot of the results for the user to view.

1.3.1 Microcontroller

The purpose of the microcontroller is to control the required components of the FRA system and as well as three other fundamental purposes:

1.) Allow the system to begin and end the test

2.) Control the VFO to generate a sinusoidal waveform at a specific frequency

3.) Collects data from the gain and phase detector and transmits the data to the system interface

At the start of a test, the microcontroller calculates 200 frequency points that are equally distributed under log scale as recommended by IEEE Power and Energy Society standards [1]. During the test, the microcontroller controls the VFO to generate a sinusoidal waveform to inject into the test unit. After the output signal reaches steady state, the microcontroller reads the gain and phase values from the detector and transmits the data to the user's PC. The microcontroller will also transmit the data to a web database through Wi-Fi for long distance monitoring purposes.

1.3.2 Variable-Frequency Oscillator

The purpose of the VFO is to produce a sinusoidal waveform with a controllable frequency range of 10 Hz to 2 MHz. The output frequency range and frequency step of the VFO is controlled by the microcontroller.

1.3.3 Amplifier

The purpose of the amplifier is to increase the voltage level of the output signal of the VFO to achieve a voltage of 5 V_{peak}. A 10 V/V gain will be sufficient based on the 0.5 V_{peak} output voltage of the VFO. The frequency bandwidth of the amplifier will be at least 2 MHz to accommodate the maximum test frequency of 2 MHz. The amplifier will also ensure that adequate power is supplied to the test unit to ensure consistent measurements of the gain and phase [1].

1.3.4 Gain and Phase Detector

The purpose of the gain and phase detector system is to determine the test unit's transfer function gain magnitude and phase angle over the desired range of frequencies. The gain and phase values gathered at this step will then be sent to the microcontroller where it will be used to construct the Bode plot for the test unit.

1.3.5 System Interface

The purpose of the system interface program is to start the FRA test, store, and plot the FRA test results. The interface program will collect the data of the gain and phase from the detector, and will store the data into a comma-separated values (CSV) file. In addition, the interface program will generate a Bode plot of the test unit based on the data of the gain and the phase values.

Chapter 2

Project Overview

2.1 Project Specifications

The designed FRA system required multiple hardware components to operate. The system flow chart is shown in Fig. 1.1 and depicts how the hardware components were utilized in the flow of operation within the system. Further details of each hardware component is discussed in the component specifications below.

2.1.1 Microcontroller Specifications

The microcontroller has multiple functions and requirements as described in Section 1.3.1 Microcontroller. In order for the design to work, the microcontroller had to meet certain specifications. Four channels for input and output was required for the microcontroller to communicate with the VFO. Two analog input channels was compulsory to allow the gain and phase values from the detector to be read. The serial port allows the microcontroller to communicate with the user's PC. Table 2.I summarizes the specifications that was required for the microcontroller.

Table 2.I: MICROCONTROLLER SPECIFICATIONS

	Requirements
Digital I/O	4 Channels
Analog Input	2 Channels
Output voltage	5V DC
Communication channel	Serial Port and Wi-Fi capabilites

2.1.2 Variable-Frequency Oscillator Specifications

The VFO is used to generate a sinusoidal waveform to be injected into the test unit. The frequency range, 10 Hz to 2MHz, was determined based from the frequency test range of the project specifications. In order to obtain accurate results, the frequency must be stable to a certain degree of error as required by the project specifications. Table 2.II summarizes the specifications that was required for the VFO.

Table 2.II: VARIABLE-FREQUENCY OSCILLATOR SPECIFICATIONS

	Value	**Notes**
Output Waveform	Sinusoid	
Frequency Range	10 Hz to 2 MHz	Minimum Range
Output Voltage Level	1Vpp	Min
Frequency Stability Error	1%	

2.1.3 Amplifier Specifications

The amplifier is used to obtain the necessary voltage level and increase the power supplied to the test unit. The amplifiers frequency bandwidth will be at least 2 MHz to accommodate the maximum test frequency of 2 MHz. For a sine wave not to be affected by slew rate limitation, the slew rate (SR) will have to be larger than 2π f x V_{peak}. Based on the output requirement of the VFO, the output is 0.5 V_{peak} and highest frequency is 2 MHz, which yields a minimum SR of 6.3 V/μs. The lowest impedance of a transformer winding at a given frequency is 7Ω, thus with a voltage of 5 V_{peak}, a minimum of approximately 0.71 A is needed.

Table 2.III: AMPLIFIER SPECIFICATIONS

	Value	**Notes**
Amplifier Type	Voltage Amplifier	
Gain	10 V/V	Min
Frequency Bandwidth	2 MHz	Min
Slew Rate	6.3 V/μs	Min
Output Voltage Level	5 V_{peak}	Nominal
Output Current Level	0.71 A	Min

2.1.4 Gain and Phase Detector Specifications

The purpose of the gain and phase detector system is to determine the gain magnitude and phase angle of the transfer function of the unit under test over the desired range of frequencies. In order to produce an appropriate Bode plot, the phase angle must range from 0 to 180 °. The gain and phase detector specifications are listed below in Table 2.IV.

Table 2.IV: GAIN AND PHASE DETECTOR SPECIFICATIONS

	Value	**Notes**
Phase angle range	0 to 180 degrees	Min

2.1.5 System Interface Specifications

The system interface program is required to perform multiple functions in order for the design to operate. The programming language was be chosen such that the development difficulty level is not beyond our capabilities. The system interface must be functional on different operating systems to allow flexibility for the user. The Processing development environment was chosen to build the interface program. The required functions of the system interface program are listed below in Table 2.V.

Table 2.V: SYSTEM INTERFACE SPECIFICATIONS

	Requirements
Development Environment	Processing IDE
Programming Language	Processing
Operating System	Mac OS X, Windows
Function 1	Start FRA test
Function 2	Data Storage to CSV format
Function 3	Bode plot

Chapter 3

Research, Part Acquisition, and Draft Design

3.1 Microcontroller

The research for microcontroller began in May 2014 by all group members. The idea was to find an embedded computing system that would meet our project specifications, be able to control all the necessary components and be easily programmable. In order to choose the platform properly, all the interfacing requirements for the microcontroller were based on other component selections. The general types of required interface channels are listed in Table 3.I with their associated project component.

Table 3.I: UPDATED MICROCONTROLLER SPECIFICATIONS

Project Component	**Interface Channel**
VFO	4 parallel digital output
Amplifier	N/A
Gain and Phase Detector	2 parallel analog input
Interface Program	Serial port & Wi-Fi channel

The Arduino Uno was chosen as the first microcontroller. The Uno has 14 digital I/O channels that enables control of the other system components [2]. The UNO has an internal analog-to-digital converter (ADC) with four I/O channels to convert an analog signal to digital data. The core of the Arduino is based on JAVA, which is within our capabilities to program. The Arduino Uno was unit tested (detail will be discussed in Section 4.1),and was able to control the VFO. The analog input function of the UNO was working effectively as well. Since the Arduino Uno was working to our satisfaction, it was decided to add the wireless communication function to the microcontroller system. The Arduino Yun and Uno share the same motherboard which meant the code used did not require any adjustments, the only difference is that the Yun has a built-in Wi-Fi module [3]. By using the Arduino Yun, the system was able to communicate with a web-based database wirelessly and plot the data on a public website. The updated microcontroller system design with the Arduino Yun is shown in Figure 3.1

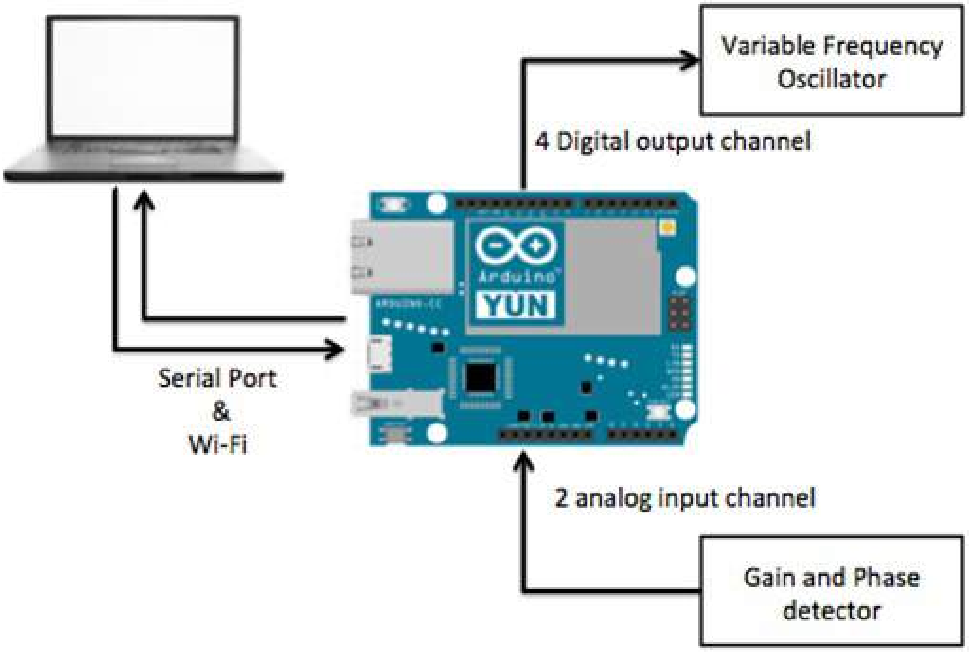


Fig. 3.1: Updated Microcontroller design with the Arduino Yun

3.2 Variable-Frequency Oscillator

The research for VFO was completed in conjunction with the microcontroller. The AD9850 was selected because it meets all project specifications and is superior to the other options available. In addition, the AD9850 is fully compatible with Arduino based devices [2]. The AD9850 also has a very high frequency resolution allowing accurate control of the output frequency which is a necessity for the purpose of the project [4]. The AD9850 VFO is shown in Fig. 3.2

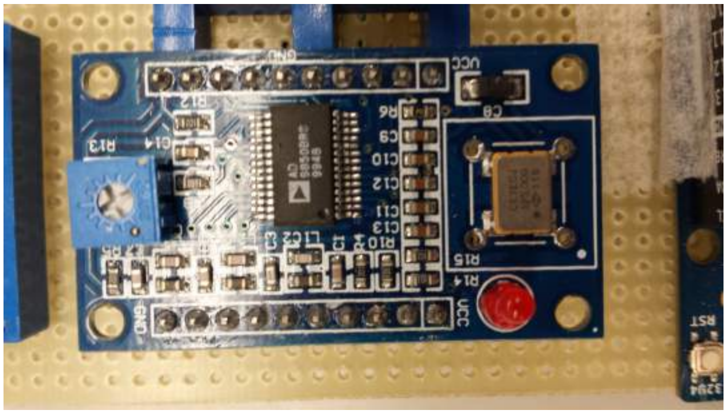


Fig. 3.2: The AD9850 VFO

3.3 Amplifier

The research for an amplifier was conducted by all the group members in July 2014. Research was conducted on both voltage and power amplifiers according to to the specifications in Section 3.3. The choice of the amplifier was made in conjunction with the VFO unit test results (Section 4.1) because the slew rate had to be determined from the peak voltage of the VFO. The VFO unit test obtained a perfect sinusoidal waveform of 1.04 V peak-to-peak value. For a sine wave not to be affected by slew rate limitation, we needed an amplifier with a slew rate of:

$$\mathrm{SR} = 2\pi \mathrm{f}\ \mathrm{V}_{peak} = 6.5\ \mathrm{V}/\mu\mathrm{s}$$

To obtain a peak voltage of 5 V, the gain of the amplifier had to be $\geq$ 10 V/V. After conducting research, two amplifiers were found, Ultra-Wideband, Current-Feedback Operational Amplifier with Disable (OPA695) and GaAs PHEMT MMIC Power Amplifier (HMC659). OPA695 is an operational amplifier with a 450MHz bandwidth and +8 Gain [5]. Whereas the HMC659 is a power amplifier with an output power 26.5dBm and the Gain is 19db [6]. After consulting with Daniel Card, he recommended not to use both of the amplifiers because radio frequency amplifiers generally work on a fixed output and input impedance level so that they can drive matched transmission lines (for maximum power transfer and minimal distortion). After performing more research, the High-Voltage, Low-Distortion, Current-Feedback Operational Amplifiers (THS3091/5) was found. The THS3091 has a slew rate of 7300 V/μ and a bandwidth of 210 MHz [7]. The gain is 10 V/V depending on the configuration used [7]. These requirements exceed the project needs and resulted in the THS3091 being chosen as part of the amplifier design.

3.4 Gain and Phase Detector

The research for the gain and phase detector began in June 2014 by all members of the group. The research consisted of reading written literature and articles to explore the different possibilities to complete this task. The first design idea consisted of reading the analog input and output signals of the test unit by a microcontroller and storing these values onto a memory storage device. Then by using a computer program, such as Matlab, to go through the data collected to calculate the gain magnitude and phase difference of the corresponding signals. This method was the most straight forward design at the time but it was determined to be too time consuming because of the all the data points that need to be collected for the 200 different test frequencies, the time needed to allow the waveform to stabilize and finally the calculation time for the program. Another complication with this initial design would be to find a microcontroller or ADC that will have a fast enough clock speed to sample the signals and store the values.

The research eventually led to the AD8302 RF/IF Gain and Phase Detector, which is a fully integrated chip that can automatically detect the gain and phase difference of the input reference signal and secondary test signal provided to it. The output of the AD8302 chip would be the magnitude of the gain and the phase difference, both values in the form of a scaled DC voltage level. The gain and phase voltage levels would be translated to dB and degrees by the following equations:

$$V_{MAG}[V] = 0.03[V/dB] \times Gain[dB] + 0.9[V] \tag{3.1}$$

$$V_{PHS}[V] = 0.01[V/^{\circ}] \times (\pm Phase[^{\circ}] + 180^{\circ}) \tag{3.2}$$

These equations are obtained from the AD8302 data sheet [8]. The AD8302 chip was chosen for our gain and phase detector subsystem design because it would simplify the design by allowing the gain and phase detection to be encompassed into one single unit, whereas the initial idea required multiple components to achieve the same goal.

The choice of the AD8302 as the gain and phase detector subsystem came with some complications, one of which is the ability to distinguish between positive and negative phase angles, this issue was solved by microcontroller coding discussed in the Negative Phase Angle Subprogram (Section 5.2 . The second issue is the low frequency performance. The AD8302 is a RF/IF device and the AD8302 datasheet provides typical performance characteristics for RF/IF frequency ranges. This created a problem because the frequencies [>100 MHz] tested on the datasheet are well above the desired frequency range [$\leq$ 2 MHz] for the purpose of our project. This problem is discussed in the application note Operation of RF Detector Products at Low Frequency[9]. This application note recommends adding external circuitry, as seen in Figure 3.3, to help increase the AD8302's performance at lower frequencies [$\leq$ 1 kHz] [9].

The application note states that by increasing the coupling capacitors, C$_c$, values, the performance of the AD8302 will be improved [9]. This application note contained results for different C$_c$

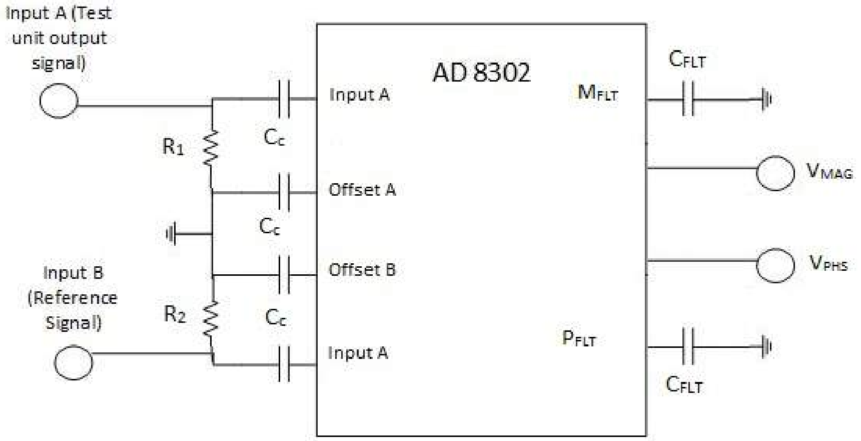


Fig. 3.3: Low Frequency External Circuit of Detector to improve performance[8].

values and graphs were made based on the conformance and the error produced by each C_c value. For the different frequencies and C_c values, the application note states that to obtain the best performance for frequencies in the 10s range, it would be recommended to choose $C_c \geq 10\ \mu F$ [9]. For the purpose of the project, it was decided to choose $C_c = 100\ \mu F$ based from recommendation and because it had the least amount of error. The capacitor values C_{FLT} are implemented to reduce the ripple in the DC voltage levels of V_{MAG} and V_{PHS} and this value was chosen to be 10 μF based from the application note because it would provide the least amount of ripple and more accurate readings [9].

Another issue with the AD8302 was the maximum allowed input voltage at V_{INA} and V_{INB} which is 1 V_{rms}. The problem occurs with the project specifications requiring a peak voltage of 5 V to be sent to the test unit which is too large for the reference signal of the detector. The solution to this problem required that an attenuator voltage divider circuit, as seen in Figure 3.4, be designed to lower the voltage level prior to the detector. The amplifier that was designed for the project produces a gain of 10 V/V and the voltage output of the VFO is 0.5 V_{peak}, hence a voltage of 5 V_{peak} was required to be attenuated to $\leq 1\ V_{rms}$.

$R_1 = 1\ \mathrm{k}\Omega$, $R_2 = 56\ \Omega$

Attenuation = 56/1056

Thus, V = 5 V_{peak} is attenuated to 0.1876 V_{rms}, which is within the operable range.

This attenuation meant that the gain measured by the detector will now be affected, this issue was taken into consideration and accommodated for by adjusting the values within the code that converts the measured DC voltage level to decibels. The attenuator will also help meet the operating specifications of the chip that requires the gain be within ± 30 dB, this issue arises when the voltage output level of the test unit is much less than the input voltage resulting in a gain less than -30 dB. By reducing the input reference voltage, it will allow for the gain to fall within the operable range of ±30 dB.

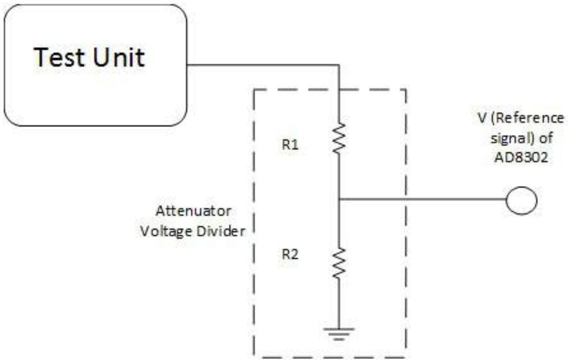


Fig. 3.4: Attenuator voltage divider circuit to meet AD8302 limitations

After receiving the device, it was found that the AD8302 chip was not compatible with the breadboard, smaller in comparison, it was required to be built onto printed circuit board (PCB) with adapter prongs to allow proper fitting onto a breadboard as seen in Fig.3.5 . This task was completed, thanks to Sinisa Janjic.

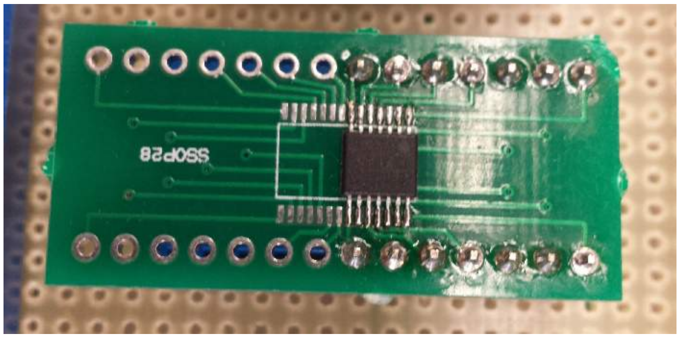


Fig. 3.5: AD3802 detector built onto PCB to allow proper fitting onto breadboard

3.5 System Interface

The system interface collects, displays and stores the gain and phase values of the test unit over the entire test frequency range. The Arduino microcontroller collects the analog signal of the gain and phase from the gain and phase detector. Through the ADC on the microcontroller, the analog signal of the gain and the phase would then be converted to digital data. The system interface then collects the digital data through serial communication with the microcontroller. The system interface visualizes the data acquisition process by real time plotting the gain magnitude and the phase on the testing frequency range in log scale. At the same time, the system interface stores the data into a CSV file for future analysis and implementation.

Our first task was to select a development environment and programming language to develop the system interface. Through research, we selected Processing IDE, which is a java-based high-level programming language and development environment. Processing was chosen for four reasons. Firstly, Processing is powerful for visualizing data because it is designed for building electronic visual art [10]. Processing has several built-in functions for drawing different shapes within a user-defined area, which would decrease the coding complexity. Secondly, Processing IDE allows us to develop and implement the system interface in both Window operating system and MAC OS operating system. Thirdly, Processing has a very good compatibility with the Arduino microcontroller. Through serial port, Processing can easily communicate with Arduino Microcontroller. Finally, the Processing is free and open source, and there are over 100 supporting libraries for developing compact Processing code.

During the research stage, we anticipated three major challenges for designing the system interface. Firstly, because the FRA test will be conducted on an unknown two-port network, the gain and the phase difference will vary randomly for different test units. To maintain a good resolution for the final graphs, we cannot use a stationary scale system for the graph. The scale has to be automatically generated by the maximum value of the gain magnitude and the phase difference. For a real-time plot, each time the system interface reads a new value of the gain and the phase, this value has to be compared with the previous value to determine the maximum value among all the data. The generation of the scale has to be adjusted each time a new value comes into the system interface program. Secondly, the microcontroller will write the gain, the phase difference and the frequency on the serial port at the same time. The system interface will read these data from the serial port. To successfully plot and store the data, we have to determine an effective way to separate three variables of the gain, the phase difference and the frequency and store them into three different data holder in the code for further implementation. Thirdly, to store the data into a CSV file, the system interface should be able to create a CSV file at a user preferred location and store the data into the each row of this CSV file. Each time a user starts a new FRA test, the system interface should be able to update the CSV file with the new data without overwrite the history data from previous test. The detail solutions to the above problems will be discussed in the section 5.5 System Interface coding.

3.6 Online Data Analysis Program (ODAP)

The online data analysis program (ODAP) completes the same task as the system interface program, except the ODAP displays the FRA plot on an online database wirelessly. In the beginning, we attempted to establish a wireless channel between Processing and the Arduino. The connection was established successfully, but this connection was very unstable. The transmitted packets had a loss rate of over 12%. It was then decided to use cloud technology to avoid this unstable communication. The ODAP is developed base on Plotly. Plotly is an open source cloud database, which allows the user to upload their data to a cloud database and real-time plot the data on a public website [11]. By embedding a small program in the Arduino Yuns root directory, the Arduino can upload data directly to Plotlys online database. The user is also allowed to manipulate the data and the plot diagram on Plotlys online GUI website. By using Plotly, we now have two sets of interfaces. One is on the operating computer, which is for FRA test technician; its function has been described in Section 3.5. The other one is online which is for the persons analyzing the data. This online interface program provides a reliable and efficient solution that allows access to the FRA test data without going into the test field.

Chapter 4

Component Testing and Simulation

4.1 Microcontroller and VFO

In the FRA system, the VFO is controlled by the microcontroller, thus their unit tests were combined. There were two major goals for the unit test, the first goal was to determine the signal stability error of the AD9850 (VFO) for 10 Hz to 2 MHz, and the second goal was to test the Arduino's ADC module and to obtain its accuracy. The test set-up block diagram is shown in Fig. 4.1. The unit test procedure was completed in the following steps:

1. Connect the Arduino Uno with the AD9850, use an oscilloscope to monitor the output of the AD9850.

2. Measure the period for each sinusoidal waveform generated by the AD9850 and calculate the frequency stability error.

3. Connect the AD9850 output to the Arduino Unos analog channel, use the Arduino IDE to monitor this output signal and display it in Excel.

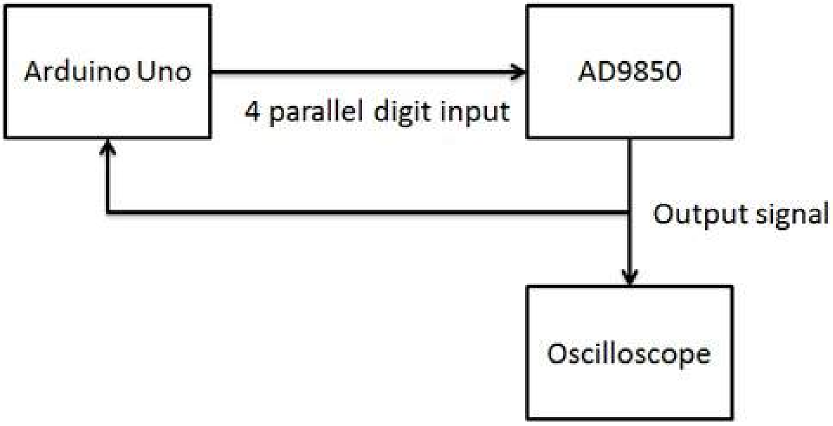


Fig. 4.1: Microcontroller and VFO unit test set-up.

The first set of test results are shown in Table 4.I, the test frequency range is from 10Hz to 10MHz. From these test results, it can be seen that the output signal amplitude is very stable up to 10MHz. The maximum frequency stability error is about ± 0.2% which is acceptable.

Table 4.I: VFO TEST RESULTS

Frequency [Hz]	**Measured Frequency [Hz]**		**Stability Error [%]**	**Amplitude [V]**
	Min	Max		
10	10	10	0%	1.04
100	99.9	100.1	+/- 0.1%	1.04
1k	999.9	1000.1	+/- 0.01%	1.04
10k	10k	10.02k	0.2%	1.04
100k	99.8k	100.2k	+/- 0.2%	1.04
1M	.999M	1.001M	+/- 0.01%	1.04
10M	9.96M	1.001M	+/- 0.4%	1.04

The second test results are shown in Figure 4.2, the complete results are in Appendix C.1. In this test, the AD9850 was used to generate a 5Hz sinusoidal waveform that was then injected into the Arduino Uno. By using the ADC of the Arduino Uno to sample the sine wave, the signal was recreated to ensure that it would be identical to the original waveform. From Fig. 4.2, the test results show that the analog input signal is a sine wave with some distortion on the peaks. The Arduino Uno datasheet states that the Uno takes 100 μs to read an analog input value [12]. This hardware limitation reveals that an accurate sample of a 5Hz sine wave cannot be properly obtained. Thus, the distortions in the test results are acceptable.

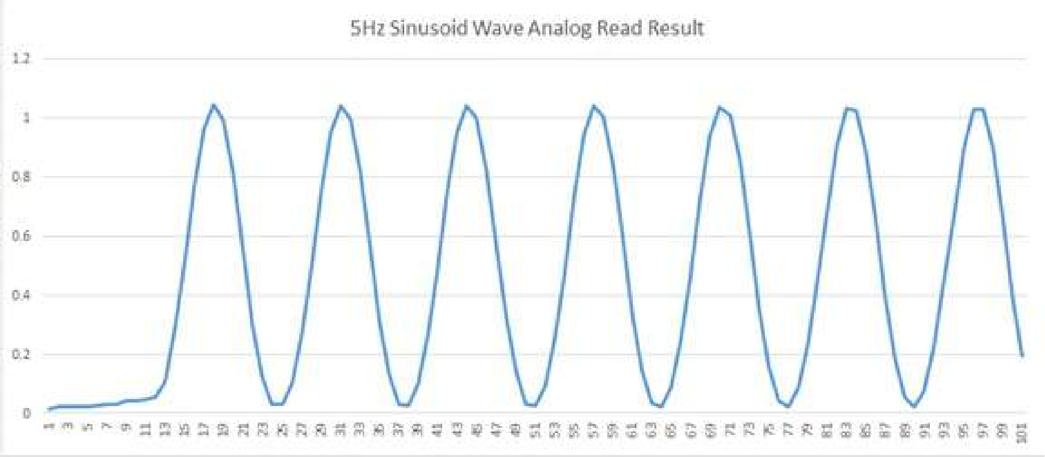


Fig. 4.2: VFO analog test results reproduced with Microsoft Excel

According to the obtained results, we can calculate the accuracy of the Arduino Unos ADC. By using an oscilloscope, the peak voltage is 1.04V, from the test results the peak voltage is about 1.09V. By using two pull-down resistors, as seen in Fig. 4.3, and reprogramming the analog read function, the error was decreased to 2%. A pull-down resistor was used to discharge static from the analog input channel and ensure that the Arduino Uno will read 0 when there is no input signal provided to it.

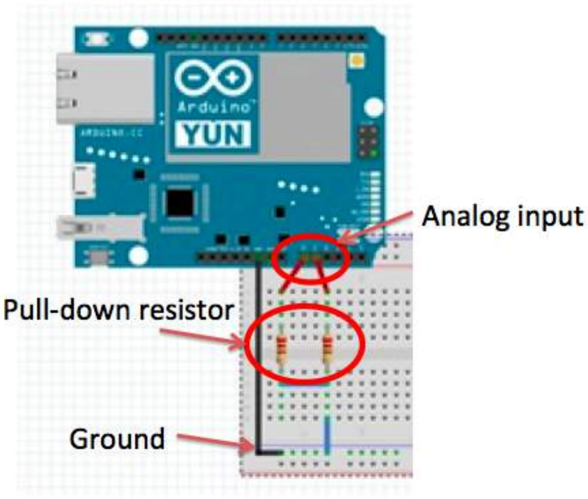


Fig. 4.3: Arduino Yun with pull-down resistors to improve the accuracy of the ADC readings

4.2 Amplifier

Different designs of the amplifier were completed on Multisim including Wideband non-inverting amplifier configuration and Push-Pull Fet amplifier configuration, shown in Fig. 4.4 to try to obtain the necessary gain of 10 V/V. The first design and simulation consisted of the Wideband non-inverting amplifier configuration. The recommended resistor values, $R_G = 95.315\Omega$ and $R_F = 866\Omega$, were chosen based from Table 4.II that was obtained from the THS3901 datasheet [7]. The amplifier was simulated with a ±15V power supply and an input voltage of 1 V_{rms}, but the desired gain was not achieved.

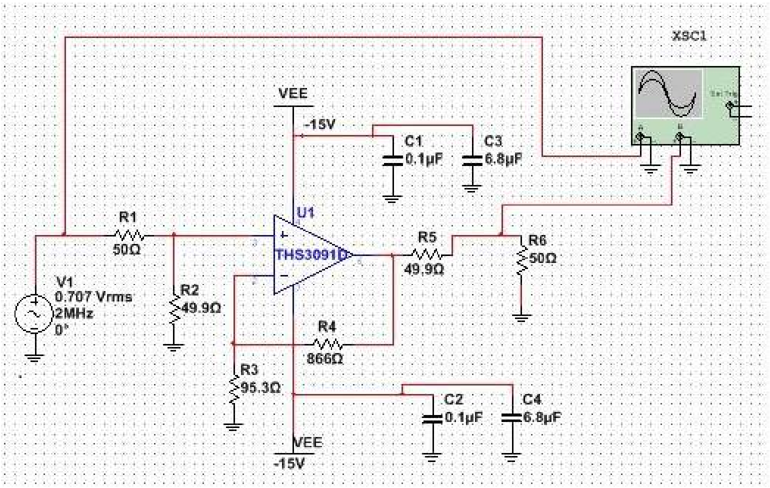


Fig. 4.4: The Multisim simulation of the initial design consisting of a Wideband non-inverting amplifier configuration

Table 4.II: THS3091 CHARACTERISTICS [7]

Gain (V/V)	**Supply Voltage (V)**	**$R_G\Omega$**	**$R_F\Omega$**
10	±15	95.3	866

The second design consisted of 4 parallel THS3091 Push-Pull Fet amplifier configurations, as shown in Fig. 4.5. A gain of 0.5 V/V was achieved because the input impedance of the four amplifiers was not taken into consideration and this caused the four amplifiers to be seen in parallel by the input which effectively reduced the gain to an undesirable amount.

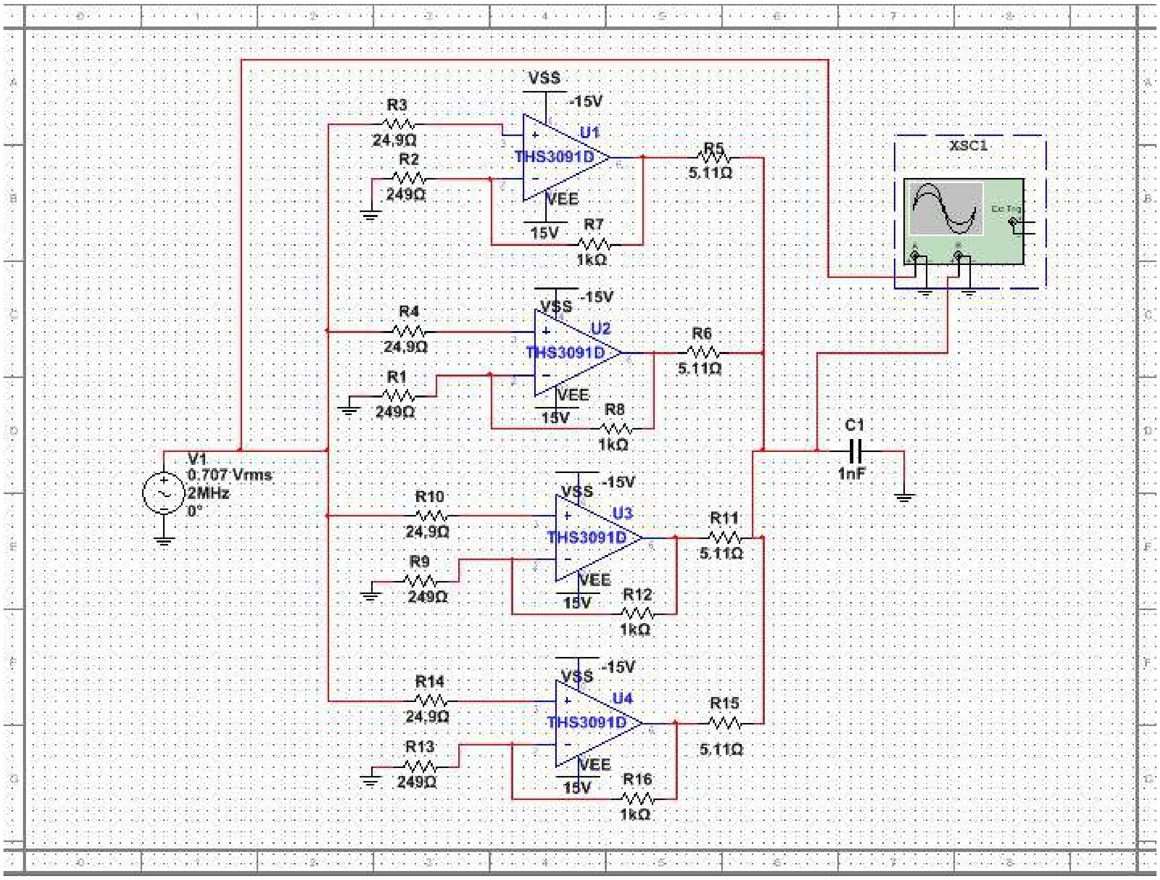


Fig. 4.5: The Multisim simulation of the second design consisting of 4 parallel THS3091 Push-Pull Fet amplifier configurations

It was decided that in order to achieve the maximum gain, it would be best to redesign the amplifier configuration to 2 parallel cascaded amplifiers. The design that was created consisted of 2 parallel THS3091 Push-Pull Fet configurations in cascade with 2 parallel THS3091 Push-Pull Fet configurations. This proved to be a success because this design met the gain requirements.

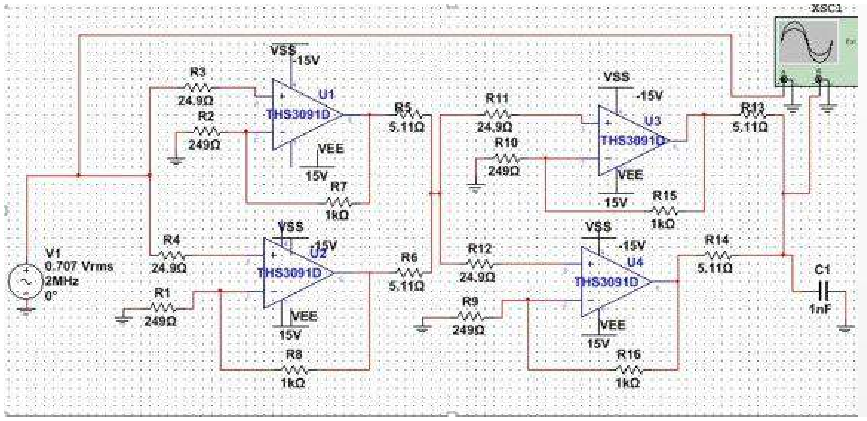


Fig. 4.6: The Multisim simulation of the third design with 2 parallel THS3091 in cascade with 2 parallel THS3091. Both stages in Push-Pull fet configurations.

The two parallel THS3091 cascade design was simulated on Multisim as seen in Fig. 4.6]. The simulation results are shown in Fig. 4.7 and it can be seen that the Gain is over 15 V/V.

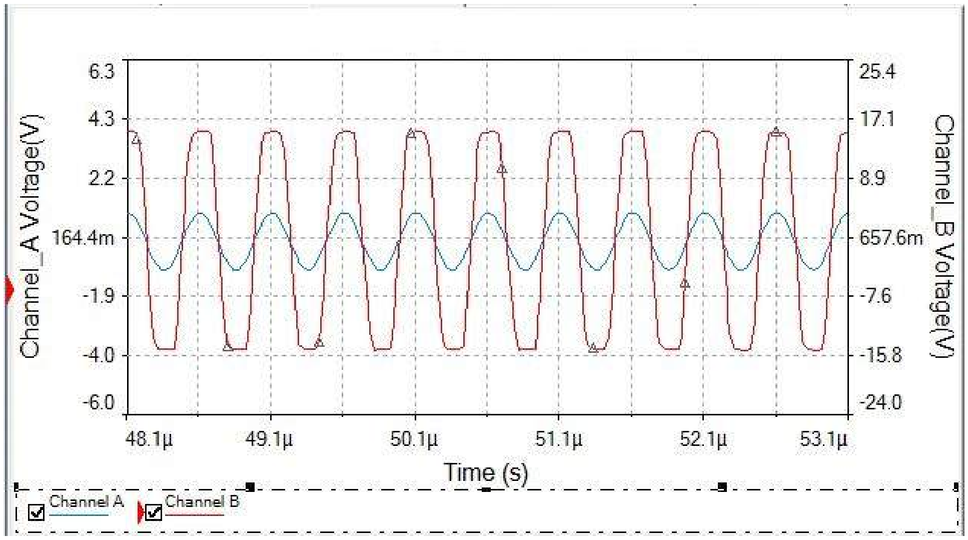


Fig. 4.7: The Multisim results, with gain over 15 V/V, of the third design with 2 parallel THS3091 in cascade with 2 parallel THS3091. Both stages in Push-Pull fet configurations. Channel B is output and Channel A is input.

This design was chosen because it met the project specifications. Luckily, an evaluation board with four THS309X chips built onto PCB was found that was very similar to this design. The evaluation board amplifier was chosen for the amplifier subsystem because it will simplify the design by eliminating the process of building our design. The amplifier found consisted of a coaxial cable input and output, and the power supply. The coaxial cables required adapters to be made by the ECE tech shop to allow the amplifier to be compatible with the breadboard. The power supply also required an adapter to be made to allow wires to be fitted. The amplifier can be seen in Fig. 4.8.

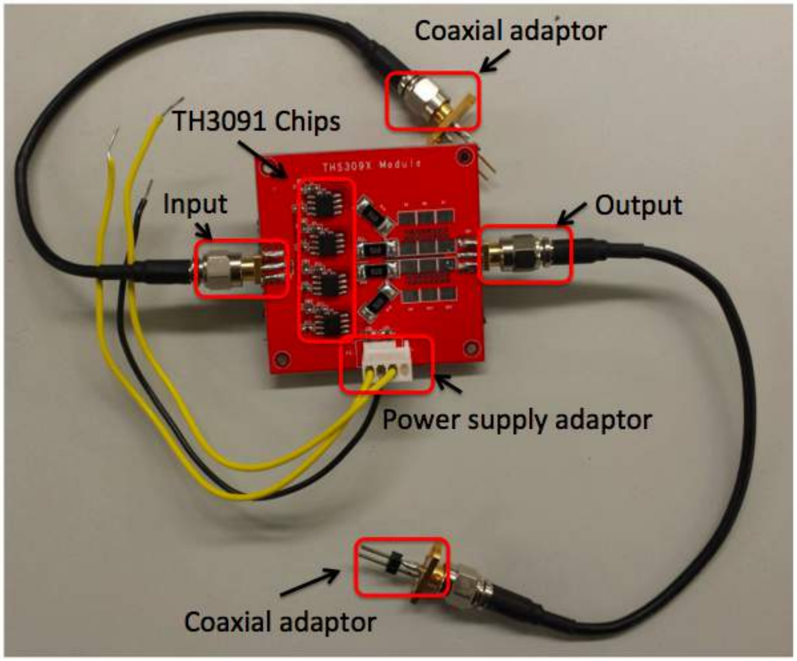


Fig. 4.8: Amplifier module with coaxial cable adapter and the power supply adapter.

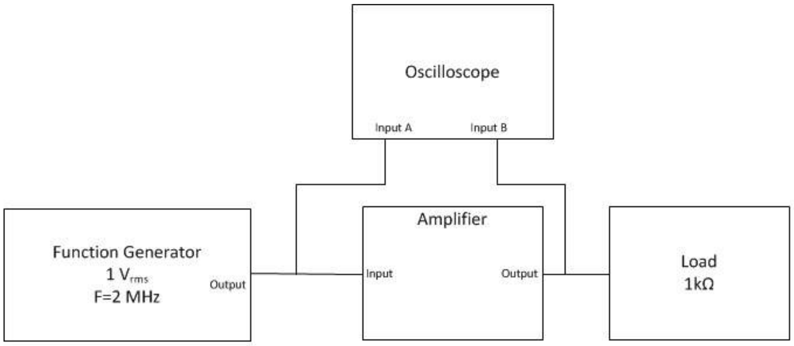
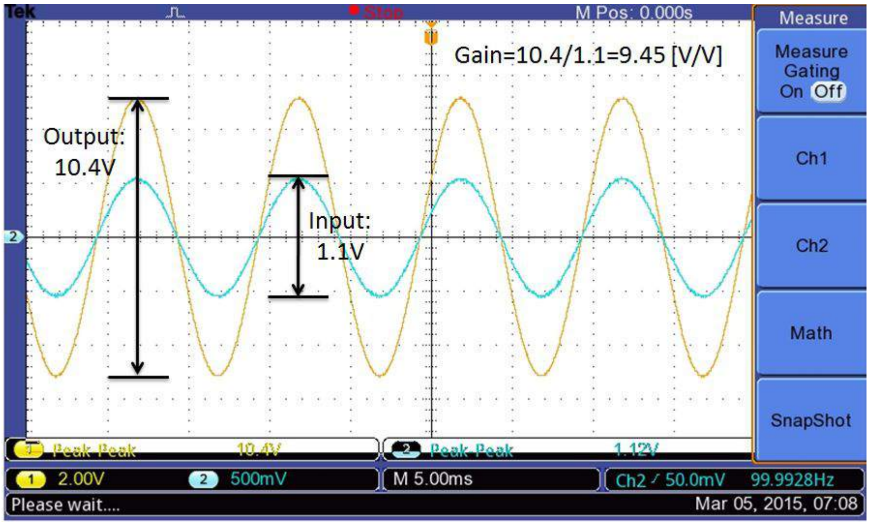


Fig. 4.9: Amplifier unit test set-up block diagram.

After receiving the amplifier at the end of November, it was unit tested to confirm that it met the project specifications. Fig. 4.9 displays the test set-up block diagram. A function generator was used to inject a 1 V_{rms} waveform, at 2 MHz, into the amplifier and the corresponding output was measured using an oscilloscope. The corresponding results from the oscilloscope are shown in Fig. 4.10. From Fig. 4.10, tt can be seen that the gain is around 10 V/V, there is no distortion and is stable for the desired frequency bandwidth.



1 Peak Peak 10.4V
2 Peak Peak 1.12V
1 2.00V 2 500mV M 5.00ms Ch2 ∕ 50.0mV 99.9928Hz
Please wait.... Mar 05, 2015, 07:08

Fig. 4.10: Amplifier test results displaying the output waveform (yellow) and input waveform (blue).

4.3 Gain and Phase Detector

Before testing began, the detector was assembled with the low frequency external circuit onto a breadboard. The first test was completed to verify that the unit received was fully functional and that the values of the gain and phase were accurately determined by the AD8302. This was completed by measuring the inputs, V$_{INA}$ and V$_{INB}$, and then calculating the gain magnitude and phase difference. The input gain and phase values were then compared with the detector outputs, V$_{MAG}$ and V$_{PHS}$. The test set-up block diagram is shown in Fig. 4.11.

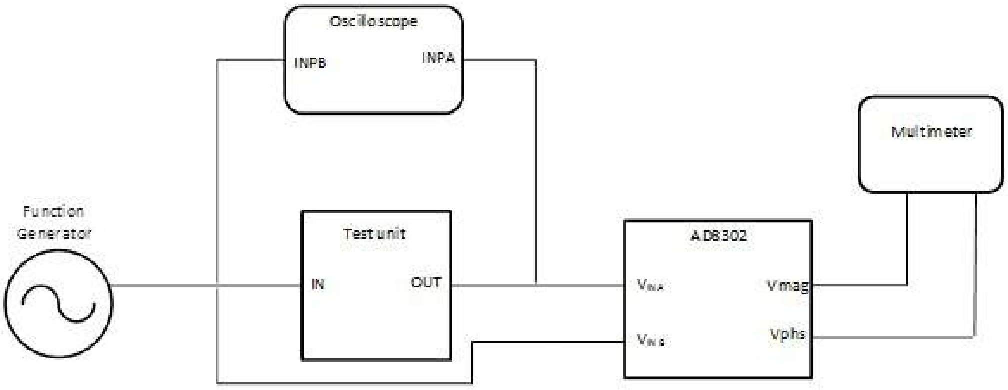


Fig. 4.11: Gain and Phase Detector unit test block diagram.

The unit test procedure was completed in the following steps:

1.) Set the function generator to output a 0.5 V$_{pp}$ sinusoidal waveform at 100 Hz to be injected into the input of the test unit.

2.) Use the oscilloscope to measure peak values of both waveforms and then calculate the gain.

3.) Determine the phase by using the oscilloscope to measure the time difference of the zero crossings of the signals on the display and then convert this value to phase difference.

4.) Measure the detector outputs, V$_{MAG}$ and V$_{PHS}$, with the multimeter. Convert these values into dB and degrees with Equations 3.1 and 3.2.

5.) Compare the two measured values

The test results are shown in Table 4.III. The gain values were found to have an average error of 0.444 dB, which is the expected error in accordance with the application note that states an accuracy of 0.5 dB was determined with the same C_c value [9]. The phase error, shown in Table 4.III, was found to be too large and was caused by the method used to determine the phase. This required that a third test be performed with a new method to determine the phase.

Table 4.III: UNIT 1 TEST RESULTS

	Input of AD8302		Output of AD8302		Error	
Load Conditions	Gain [dB]	Phase [°]	Gain [dB]	Phase [°]	Gain error [dB]	Phase error [°]
No load	0	0	0.563	12.5	0.563	12.5
Resistor [27 kΩ]	-14.54	0	-14.33	3.9	0.21	3.9
Capacitor [10 μ F]	0.6685	25.2	0.1085	16.18	0.56	9.02

The second unit test was performed to verify the accuracy of the phase measurements because the results from the first unit test for the phase values were inaccurately determined. This was completed using the same test set-up, shown in 4.11. This test was performed at different frequencies on a RC low-pass filter test unit. The values chosen were R = 3.3 kΩ and C = 0.1 μF, resulting in a cutoff frequency of 482.28 Hz. By choosing the test frequencies at 100 Hz, 500 Hz, and 1 KHz, it will ensure that the phase angles are different for each corresponding frequency. The unit test procedure was completed in the following steps:

1.) Set the function generator to produce a 1 V_{rms} sinusoidal waveform at the desired frequency to be injected into the input of the test unit.

2.) Use the oscilloscope to view the waveforms and then export the sampled waveform data onto an external USB flash drive.

3.) Measure the detector outputs, V_{MAG} and V_{PHS}, with the multimeter. Convert these values into dB and degrees with Equations 3.1 and 3.2.

4.) Determine the gain and phase values at the input of the chip by using Matlab and Microsoft Excel to calculate these values by going through all the collected sample points.

5.) Compare the two measured values

Table 4.IV: UNIT 2 TEST RESULTS

	Input of AD8302		Output of AD8302		Error	
Frequency [Hz]	Gain [dB]	Phase [°]	Gain [dB]	Phase [°]	|Gain error| [dB]	|Phase error| [°]
100	-15.02	10.8	-14.84	12.22	0.18	1.42
500	-18.10	46.8	-17.62	49.81	0.48	3.01
1000	-21.587	50.4	-22.383	53.02	0.796	2.62

The second unit test results are shown in Table 4.IV. The gain and phase values that were found on these tests have an error ± .47 dB and ± 2.35°. The phase error is significantly less than the first unit test and is now a more acceptable value in accordance with the application note that produced an error of ±1° [9].

It is noted that these values found in Table 4.IV do not necessarily match with the theoretical values of the RC filter alone and that is because of the loading caused by the external circuit of the detector. This issue was solved at a later date by removing the resistors R_1 and R_2 in Fig. 3.3. These resistors were altering the load because they were seen to be parallel with the test unit from the perspective of the detector. The main purpose of these tests were to verify that the gain and phase difference of the signals being sent to it would be accurately determined by the detector. This loading issue was solved after these tests were completed and the theoretical plot compared with the entire system (with the adjusted external circuit) plot is shown and discussed in Chapter 6: System Assembly and Testing.

4.4 System Interface

After the initial unit tests for hardware components were completed, an assembled system test with a prototype interface program was conducted on a low-pass filter. At this stage, the interface program was able to distinguish the gain and the phase difference of the test unit and was able to real time plot and store the data. Fig.4.12 shows the initial test result.

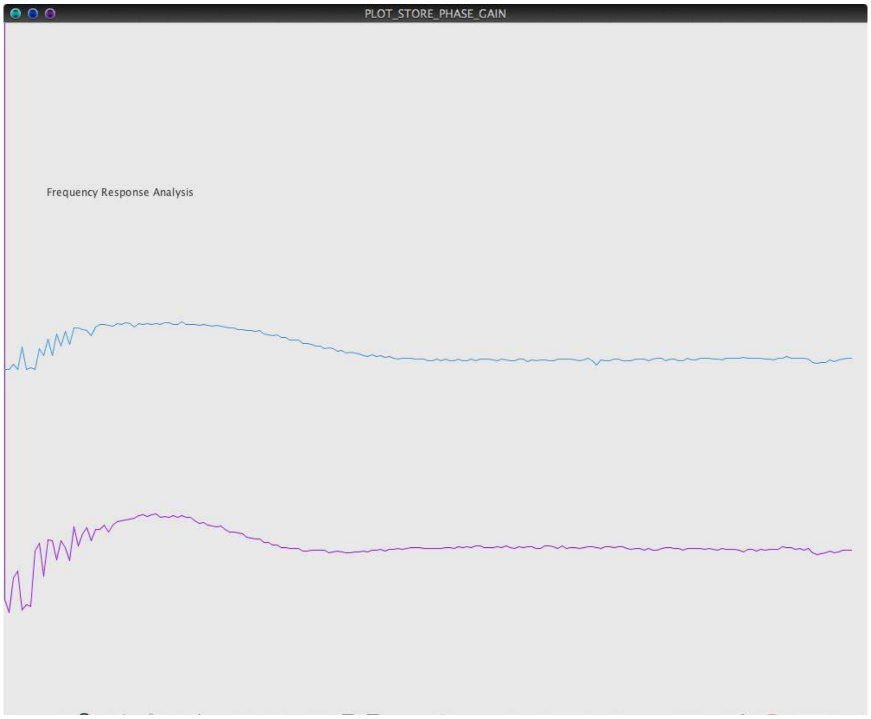


Fig. 4.12: Initial system interface test results with a low pass filter.

Through the initial test, we targeted several problems with the whole system. Firstly, the system interface did not make full use of the plotting area, which resulted in a low-resolution final graph. Secondly, at this stage, the scale for the gain and phase difference was not able to display. The Processing code for the scale was not functional with code for data acquisition. Thirdly, at the low frequency range, the values of the gain and phase difference were wrong. The wrong data led us to investigate the loading issue of the gain and the phase detector. After the first unsuccessful test, we started to investigate alternative solutions for coding the system interface. Through research, we decided to use the supporting library Grafica. Grafica is a configurable 2D plotting library for the Processing. The main features of Grafica are making linear plot in real time and automatic axis tick generation [13]. Through exploring the library examples, we completely changed the previous code for system interface in order to use the functions that the Grafica library provided. The code for serial communication part remains as before. After the gain, the phase difference, and the frequency were collected from the microcontroller; the data was plotted by the draw function provided by the Grafica library. The scale of the plot was automatically generated by the maximum value of the gain and the phase different. Fig.4.13 shows a unit test with a band-pass filter after implementing the code with the Grafica library.

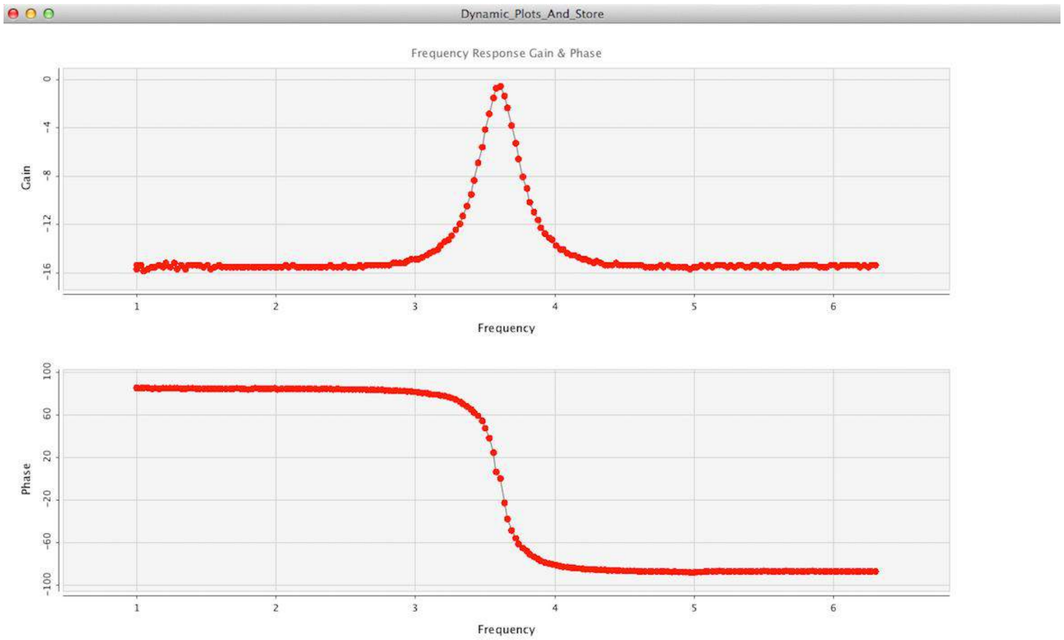


Fig. 4.13: System Interface test result with a band pass filter

As Fig.4.13 shows, the scale of the plot was generated by the maximum value of the gain and the phase difference of the test unit. In this test, the unit of the gain is in dB and the unit of the phase difference is in degrees. At this stage, we already solved the loading issue of the gain and phase detector. A new coding algorithm in the Arduino microcontroller also solved the negative angle detection issue of the gain and phase detector. As Fig.4.13 shows, the final graph for the band-pass filter has the resonance peak as expected. The phase difference varied from positive 90 degrees to negative 90 degrees. The detailed result comparison with theoretical value will be demonstrated in section 6.1. Table 4.V summarized the imported libraries that we used for the entire interface system code and their function in the code.

Table 4.V: IMPORTED LIBRARIES SUMMARY

Imported library	Library function in the code
grafica	Real time generating dynamic plot
processing.serial	Eable serial communication
java.io.BufferedWriter	Write string to a file writer
java.io.FileWriter	Write stiing to a CSV file

The detailed flow chart and explanation of the code for each function of the system interface will be explained in the Section 5.2: The System Interface Coding.

4.4.1 Online Data Analysis Program

After completing tests of the prototype interface program, we tested the ODAP on the same low-pass filter circuit. At this stage, the ODAP was able to display both the gain magnitude and phase difference on the same graph with the x-axis in log scale. The test results are shown in Fig. 4.14.

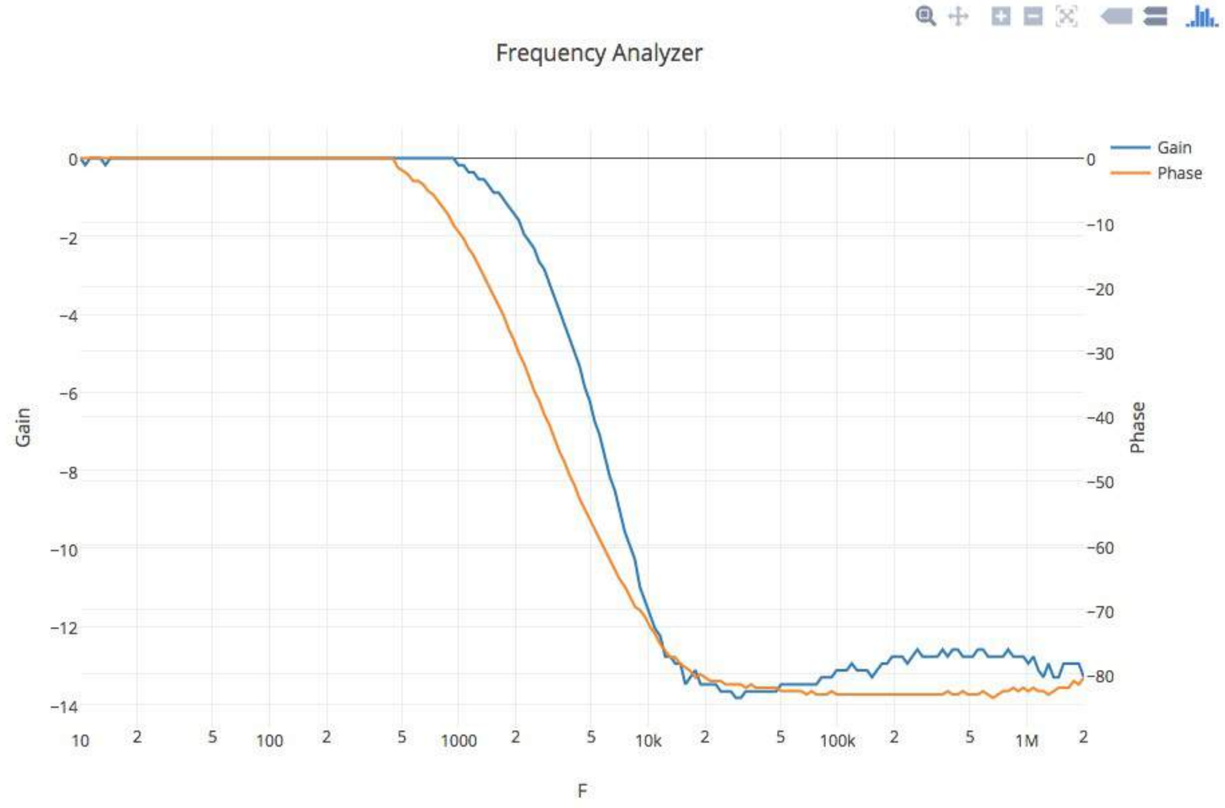


Fig. 4.14: ODAP Low-pass filter test results

As seen in Fig. 4.14, 200 data points are distributed equally by log scale between 10 Hz and 2 MHz. In this diagram, the gain of the low pass filter is shown in blue and follows the Y-axis on the left; the phase difference is shown in orange and follows the Y-axis on the right. Although there are some errors when the system is operating at high frequencies (>100 KHz), the gain and phase distribution still follows the characteristics of a low-pass filter, which indicated the ODAP was successfully operating.

After finishing the negative angle detection algorithm, the ODAP was tested on a band-pass filter. The results are shown in Fig. 4.15. From these results, it was confirmed that Plotlys Wi-Fi output channel has no interference on Arduinos serial port data transmission. When the two sets of data were exported and compared, it yielded identical values.

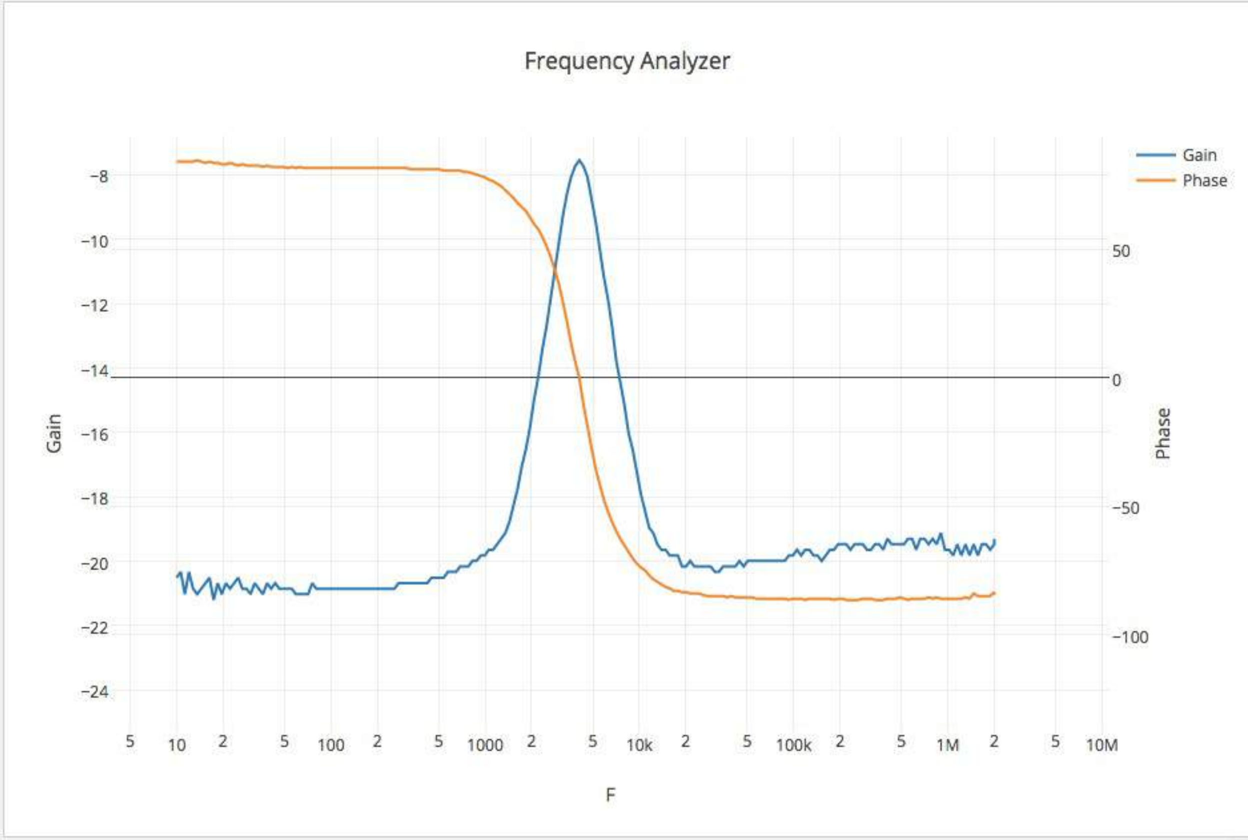


Fig. 4.15: ODAP Low-pass filter test results

Chapter 5

System Coding

5.1 Microcontroller

The main program for the Arduino is to control the AD9850 sweep frequency correctly. In the beginning of the program, the Arduino calculates the frequency power increment according to the users initial settings. By using Equations 5.1 and 5.2, the Arduino could generate 200 frequency samples that are equally distributed under log scale.

Currentfrequency: Current frequency for FRA test.

Factor: Power factor that used to calculate current frequency.

$$Currentfrequency = 10^{Factor} \tag{5.1}$$

$$Factor = log(Startfrequency) + frequencyincrement \tag{5.2}$$

After the output signal from the test unit reached steady state, the main program will allow the Arduino to start reading its analog input channels. Then the subprogram Data convention coverts all the digital values to the required values. When the Arduino collects the results, it transmits the data to the PC and a cloud database. After the current frequency reaches the end point, the main program will shut down the entire FRA system. The main program flow chart is shown as Fig.5.1.

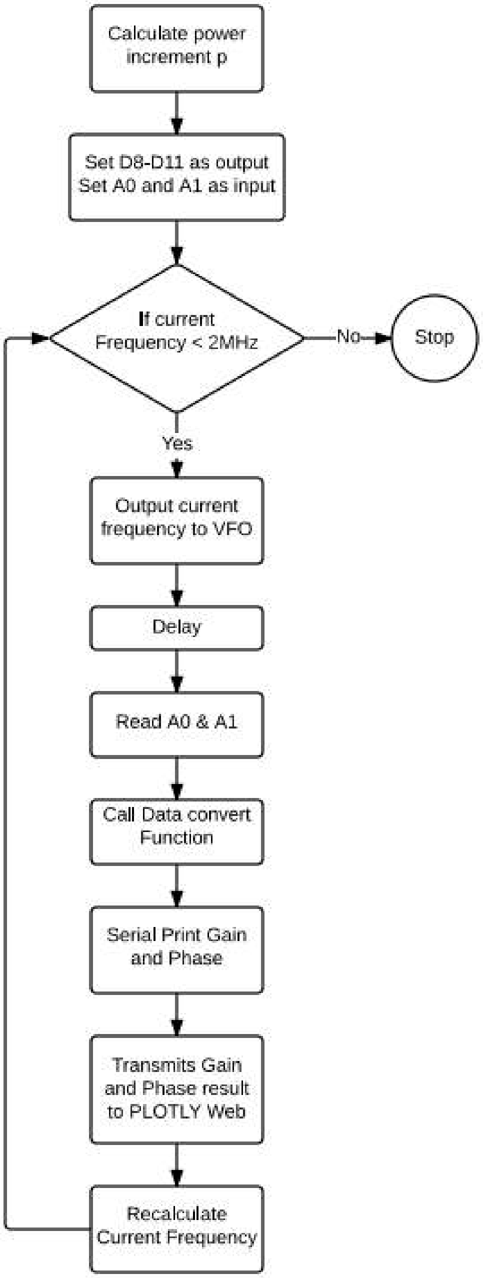

Calculate power increment p
Set D8-D11 as output Set A0 and A1 as input
If current Frequency < 2MHz
No
Stop
Yes
Output current frequency to VFO
Delay
Read A0 & A1
Call Data convert Function
Serial Print Gain and Phase
Transmits Gain and Phase result to PLOTLY Web
Recalculate Current Frequency

Fig. 5.1: Main program flow chart

5.2 Negative Phase Angle Subprogram

As mentioned before, the AD8302 cannot distinguish between positive and negative phase differences. For the requirements of the project, the test unit is a transformer, which means when the frequency is low, the circuit is inductive and when the frequency is high the circuit will become capacitive. By using this property, the problem can be solved by an embedded comparison code in our main program. Since the test unit is a RLC circuit, it can be assumed that at the starting point (f=10Hz), the phase difference is positive. When the input frequency is increased, the test circuit starts to become capacitive, which means the phase difference starts to decrease and at a certain point the phase difference becomes zero.

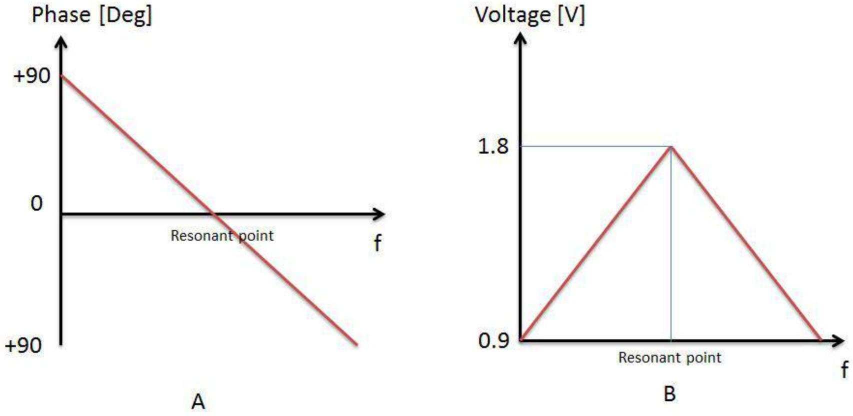


Fig. 5.2: RLC circuit phase difference tendencies used to determine negative phase angles

For normal RLC circuits, if the frequency continues to be increased, the phase difference will finally reach -90 degrees, this tendency of the phase difference is shown as Fig. 5.2A. The AD8302 phase characteristics allow this graph can be converted to Fig. 5.2B. From Fig. 5.2B, when V$_{phase}$ reaches 1.8V, it indicates that the test circuit has reached its resonant point, after that point all the phase angles are negative. Thus, the Arduino was programmed to compare V$_{phase}$ with 1.8V. Before the resonant point, a positive conversion equation to calculate phase difference was used, after that point, negative conversion equation was used to do the calculation. It is noted that due to the program's properties, this sub-program only can be used in a RLC circuit test. A detailed program flow chart is shown in Fig. 5.3.

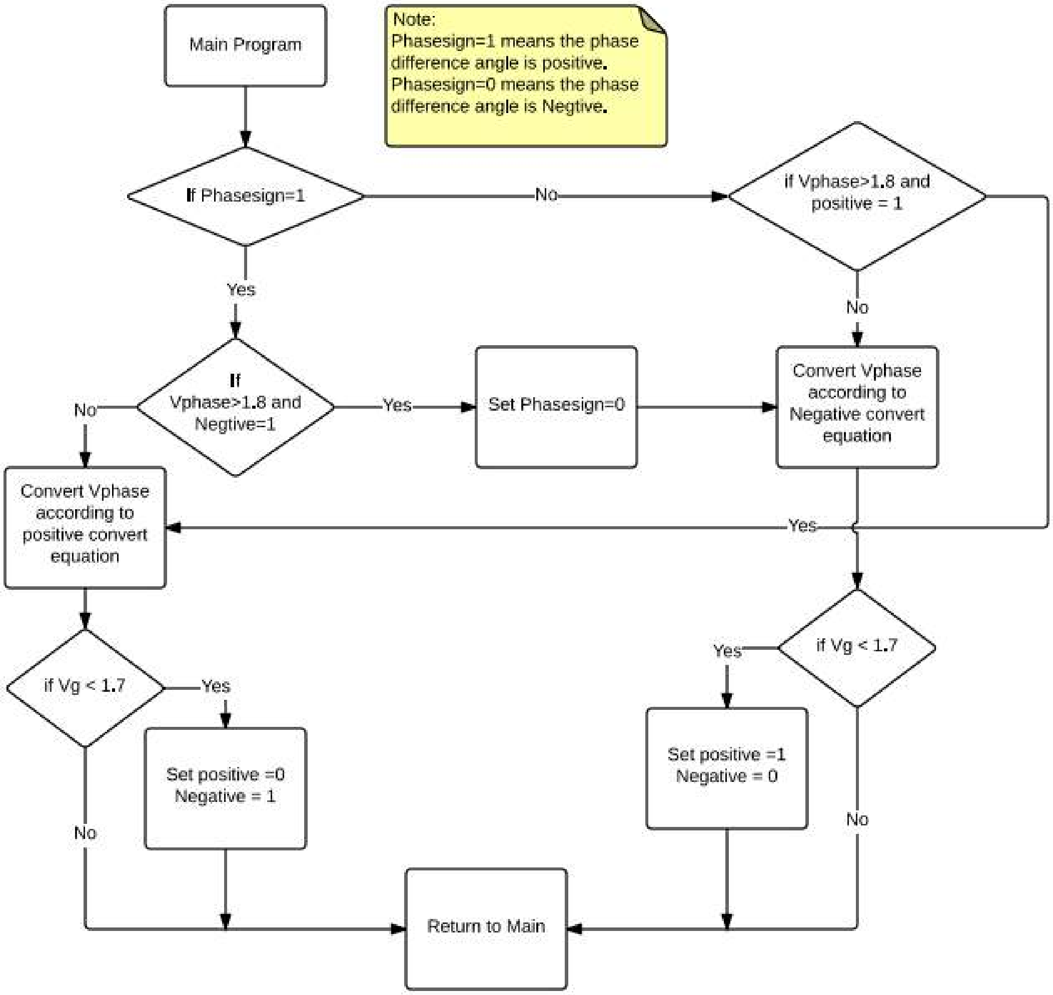

Main Program
Note:
Phasesign=1 means the phase difference angle is positive.
Phasesign=0 means the phase difference angle is Negtive.
If Phasesign=1
No
if Vphase>1.8 and positive = 1
Yes
No
If Vphase>1.8 and Negtive=1
No
Yes
Set Phasesign=0
Convert Vphase according to Negative convert equation
Convert Vphase according to positive convert equation
Yes
if Vg < 1.7
Yes
Set positive =0 Negative = 1
No
Yes
if Vg < 1.7
Set positive =1 Negative = 0
No
Return to Main

Fig. 5.3: Negative phase angle subprogram flowchart for the microcontroller

5.3 System Interface

The code for the system interface consists of four parts: reading the data from the serial port, setting up the plotting area, drawing the data and storing the data into a CSV file. In this section, the coding flow chart will be given for each part of the code.

5.3.1 Reading the Data from the Serial Port

The microcontroller writes the data to the serial port in the format of ”gain, phase, frequency”. In the system interface, the data was configured as string. So the string equals to ”gain, phase, frequency”. Then the string is split by the comma in order to separate three variables. 5.4 shows the flow chart of the reading data from serial port. After splitting the string, 3 variables are stored into an array of length three. If the length of the array is smaller than three, it means a invalid data was sending to the interface program. Finally , the data of the gain, the phase difference and the frequency are separately stored into 3 variables.

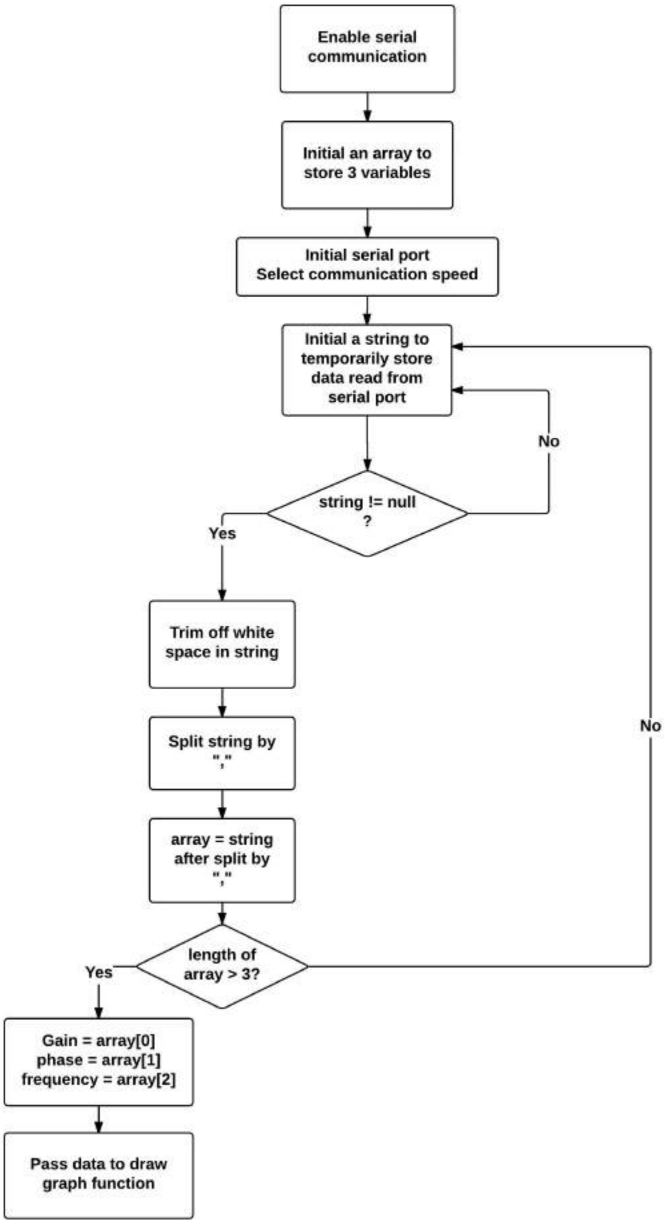

Enable serial communication
Initial an array to store 3 variables
Initial serial port Select communication speed
Initial a string to temporarily store data read from serial port
No
string != null ?
Yes
Trim off white space in string
Split string by ","
array = string after split by ","
No
length of array > 3?
Yes
Gain = array[0] phase = array[1] frequency = array[2]
Pass data to draw graph function

Fig. 5.4: Flow chart of the reading the data from the serial port for the system interface

5.3.2 Setting Up the Plotting Area

The microcontroller writes the data to the serial port in the sequence gain, phase, frequency . In the system interface, the data was configured as a string. So the string equals to gain, phase, frequency. Then the string is split by the comma in order to separate the three variables. Fig. 5.5 shows the flow chart of reading data from serial port.

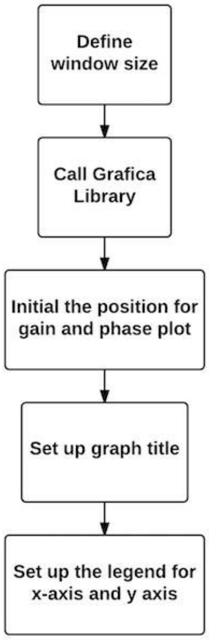

Define window size
Call Grafica Library
Initial the position for gain and phase plot
Set up graph title
Set up the legend for x-axis and y axis

Fig. 5.5: Flow chart of the reading data from serial port for the system interface

5.3.3 Data Drawing

The data drawing process was completed with the help of the Grafica supporting library. As Fig. 5.6 shows, the major function of the Grafica supporting library is to generate the scale based on the maximum value of the gain and the phase difference. The automatic scale allows us to achieve the dynamic real time plotting for ensuring the best resolution of the final graph.

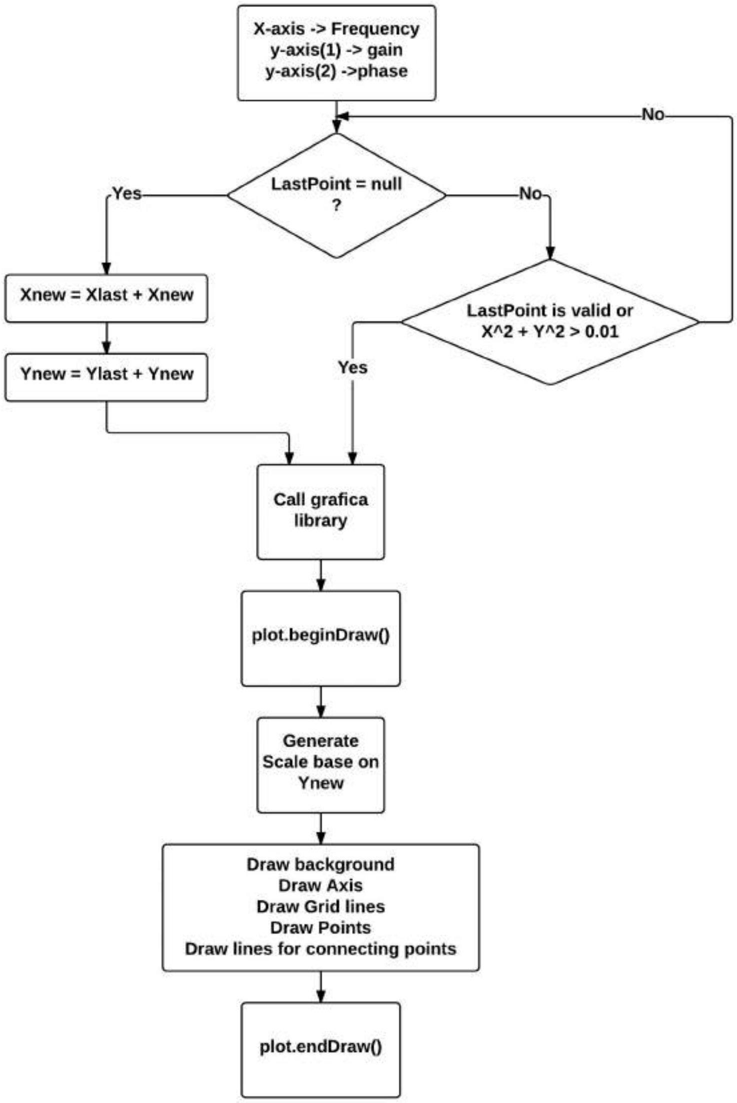


Fig. 5.6: Flow chart of drawing the data for the system interface

5.3.4 Storing Data into a CSV File

For the data storing process, we imported Java BufferedWriter and Java FileWriter. The BufferedWriter allows the program to write the data to the FileWriter. The FileWriter finally writes the data into the CSV file.

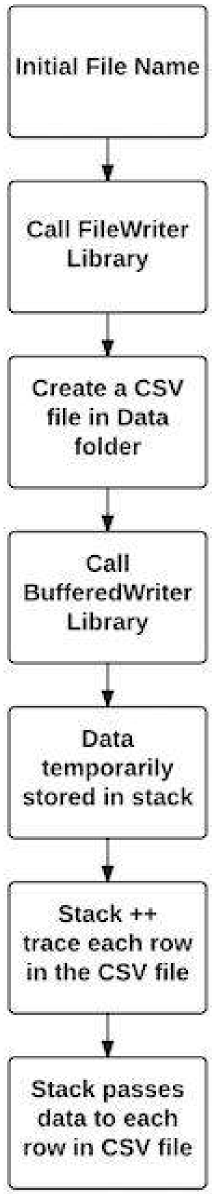

Initial File Name
Call FileWriter Library
Create a CSV file in Data folder
Call BufferedWriter Library
Data temporarily stored in stack
Stack ++ trace each row in the CSV file
Stack passes data to each row in CSV file

Fig. 5.7: Flow chart of storing data into a CSV file for the system inteface

Chapter 6

System Assembly and Testing

This chapter explains the testing that was performed in order to verify the operation of the fully assembled system. In order to improve the system's detection accuracy, the FRA system was assembled on a wire-wrap board, shown in Fig. 6.1. At the first stage, the system was tested on a low-pass filter circuit. The next test was for the system performance on a band-pass filter circuit which is almost equivalent to a real transformer. Each test entry indicates the reasoning and outline for why the test was performed, as well as the necessary preparation. In addition, the description of the steps taken in the test procedure, followed by the results that were obtained is included. Finally, each entry ends with a discussion of what conclusions were drawn based on the outcome of the experiment.

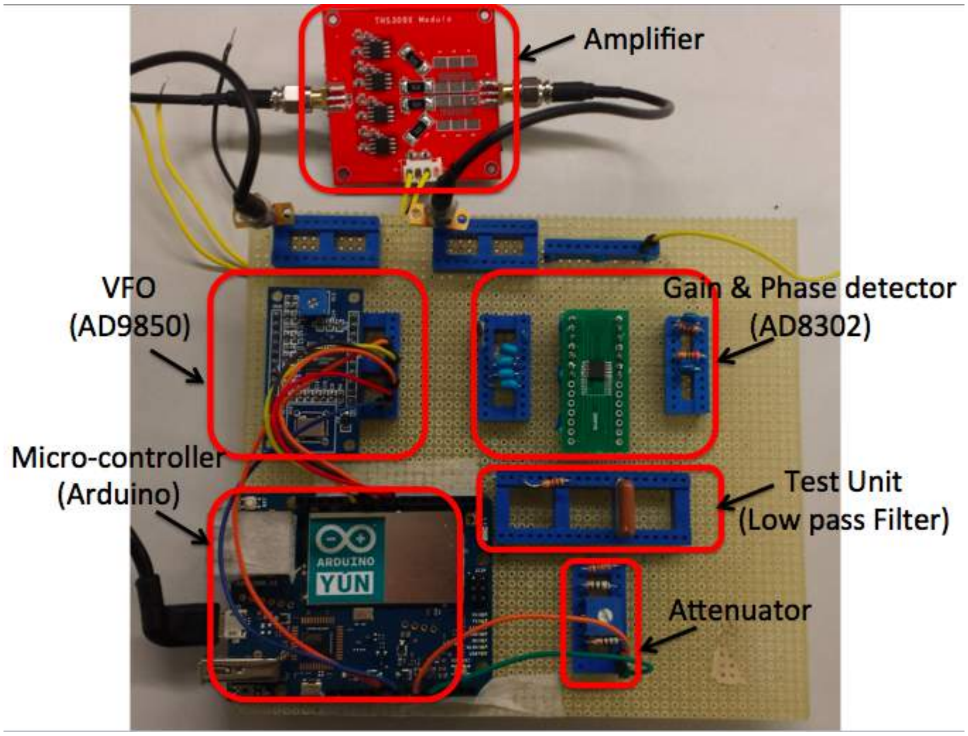


Fig. 6.1: Assembled FRA system on wire wrap board

6.1 Low-Pass Filter Test

The purpose of the low-pass filter test was to verify the performance of the designed FRA system with a first order circuit test unit. By comparing the system test results with the theoretical values, the performance of the system was observed to improve the systems accuracy. After the low-pass filter test, the system could be tested on a higher order circuit.

6.1.1 Test Setup and Procedure

The low-pass filter test set-up block diagram is shown in Fig. 6.2 and low-pass filter circuit is shown in Fig. 6.3.

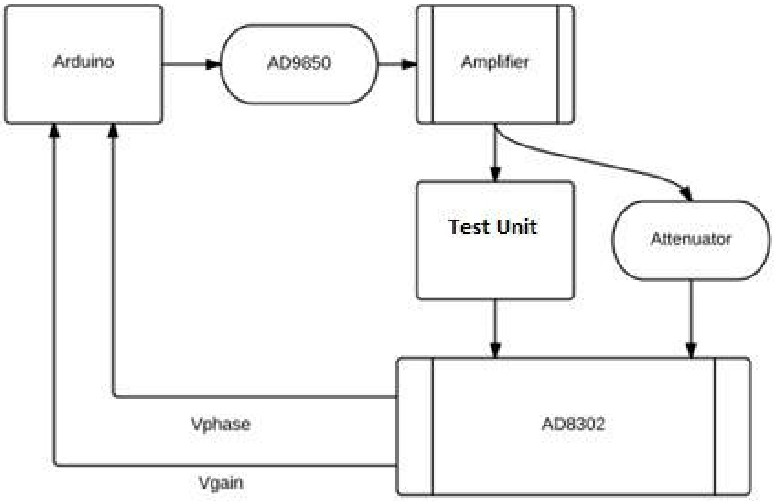


Fig. 6.2: System test set up

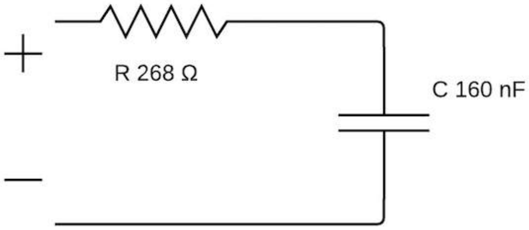


Fig. 6.3: RC filter unit used for system test

The transfer function for the low-pass filter is:

$$\frac{V_{out}}{V_{in}} = \frac{1}{1 + j2\pi fC} \tag{6.1}$$

By using Matlab, the theoretical gain and phase difference of the low-pass filter was calculated, for 10 Hz to 2 MHz, to use in comparison with the test results.

6.1.2 Test Result

The interface program can save all the data to a .csv file. The comparison results is shown in Fig. 6.4, and a more detailed table of the data can be found in Appendix C.2. From Fig. 6.4, it be seen that from the FRA test results, the phase difference was very close to theoretical values except when the test results reached -83 degrees instead of -90 degrees. For the gain, the test results were very close to theoretical value.

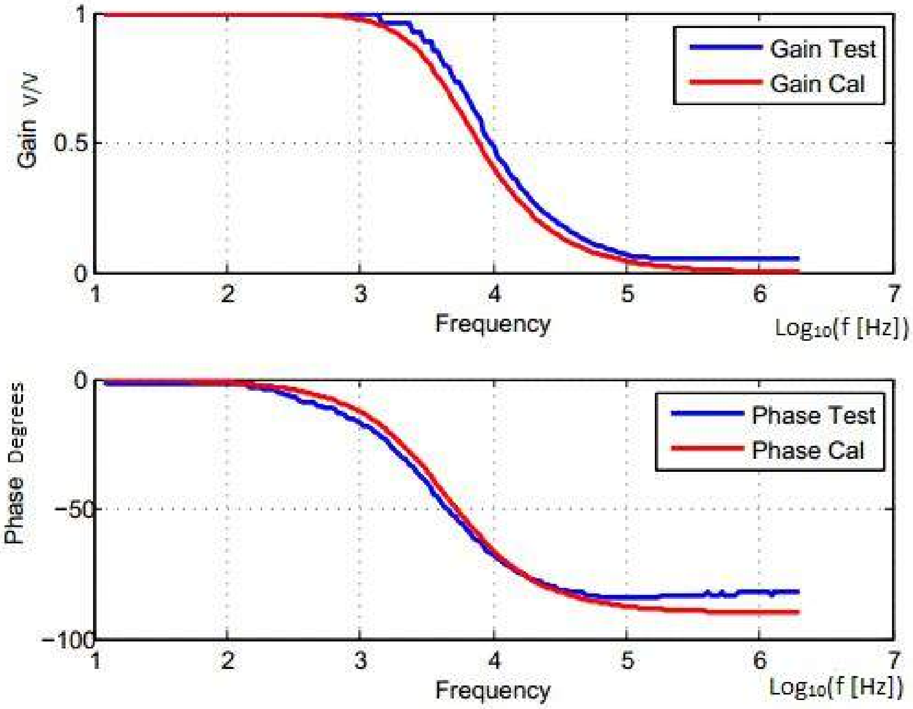


Fig. 6.4: System test reults of a RC filter

6.1.3 Low-pass Filter Test Summary

The FRA system can be used on a low pass filter because the test results were very close to the theoretical values. Although there were some small errors in the high frequency range, the results are still acceptable. After embedding the negative phase detection subprogram, our FRA system was ready for the next test stage.

6.2 Band-pass Filter Test

The band-pass filter is more closely related to the equivalent circuit of a transformer winding. Using the same procedure as the low-pass filter test, the test results from the system interface were compared with the theoretical values calculated from Matlab. By plotting the measured and calculated values in the same graph, the accuracy of the system can be seen.

6.2.1 Test Setup and Procedure

The whole system set-up can be seen in Fig. 6.2. The procedure was the same with the test for low-pass filter. The band-pass filter circuit is showed in Fig. 6.5 below.

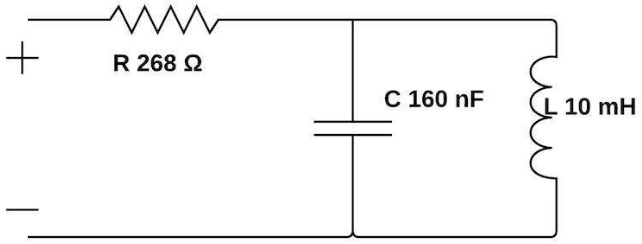


Fig. 6.5: System test results of a band-pass filter test unit

The calculated transfer function for the band pass filter is

$$\frac{V_{out}}{V_{in}} = \frac{1}{1 + jR(\omega C - \frac{1}{\omega L})} \tag{6.2}$$

Matlab was used to calculate the theoretical values of the gain and the phase of the band-pass filter over the test frequency range, 10Hz to 2 MHz.

6.2.2 Test Result

The comparison results is shown in Fig. 6.6. A detailed data table of the results can be found in Appendix C.3. From Fig. 6.6, the resonant peak frequency accurately matches with the theoretical value. The bandwidth was more narrow compared to the theoretical graph. The errors of the gain were large at the low and high frequency ranges and could be due to the resistance of the capacitor, inductor, and wires. The measured phase values were very close to the theoretical values and the final results comparison proved that the negative phase detection algorithm was functioning.

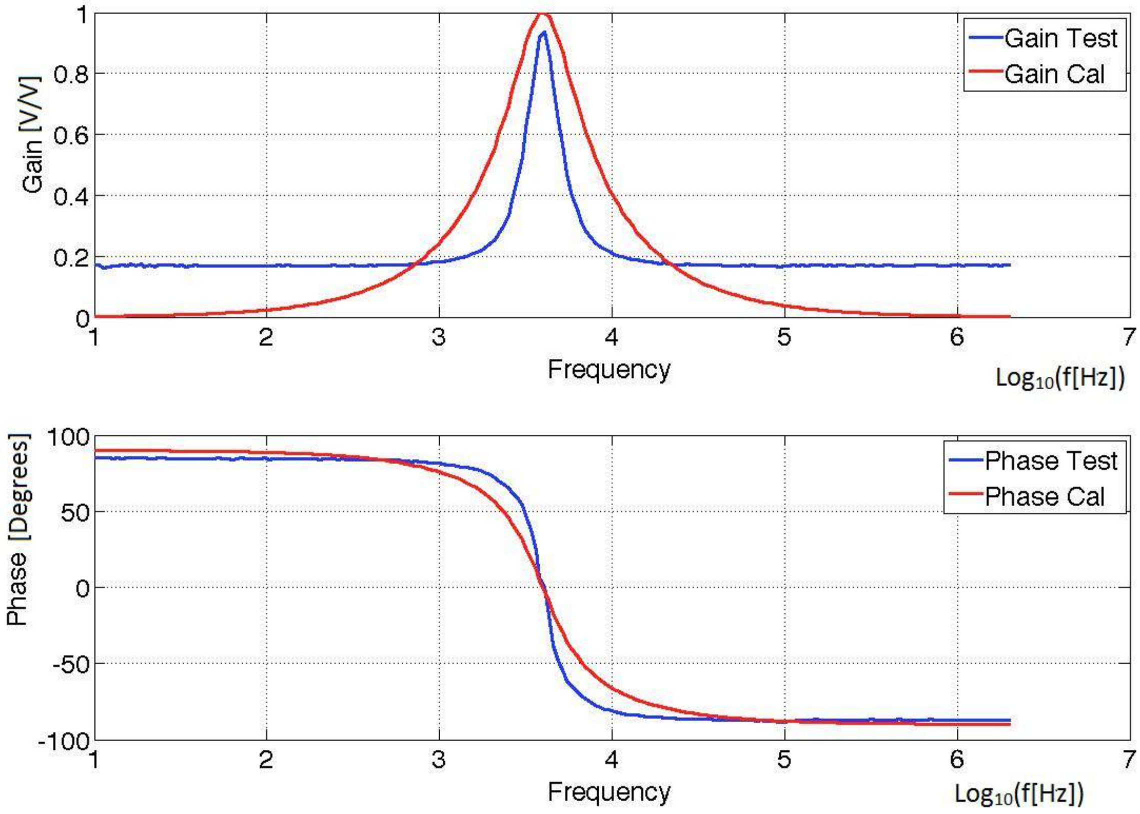


Fig. 6.6: RLC circuit comparison result

6.2.3 Band-pass Filter Test Summary

The RLC test phase difference results were very close to the theoretical values. The measured gain value had some errors compared to the theoretical values. However, the measured resonant peak frequency matched with the theoretical value. To conclude, the FRA system can be used on a RLC test circuit.

Chapter 7

Future Work and Conclusions

7.1 Future Work

The goal of this project was to determine the gain and phase angle of the transfer function of the unit under testing for a range of frequencies between 10 Hz and 2 MHz. By creating the design described, we were able to successfully meet the project goal.

As seen in Section 6.2: Band-pass Filter Test, the results were not entirely accurate. This issue could have been resolved if more time was permitted. In retrospect, the gain and phase detector initial design would have been a better option for determining the gain and phase because it will provide more accurate results if two high-speed digitizers were used instead of the AD8302 detector. Although this would have required a longer duration of time for the analysis, the results would be more accurate if the right digitizer could be found. The Arduino Yun has expandable external memory capabilities in terms of a micro-SD memory card where these results could have been stored before being transferred to the system interface to determine the gain and phase of the sampled signals. In addition, an external ADC could have been connected to the Arduino to improve the reading accuracy of an analog signal.

7.2 Conclusion

The purpose of this project was the design and implementation of a frequency response analysis system as described. Research was completed for each subsystem to allow different design considerations to be made. The different components were selected based on the required hardware specifications and compatibility with other components. After selecting and ordering each component, unit testing was completed to verify the functionality and accuracy before the entire FRA system was assembled. The system test results showed that the FRA system design was satisfactory. The low-pass filter test results were very close compared to the theoretical values. As for the band-pass filter, the error in the results is slightly greater but the resonant peak frequency is very accurate. It can be concluded that the FRA system design can be used for first and second order two-port networks.

References

[1] IEEE Power and Society [Online]. IEEE Guide for the Application and Interpretation of Frequency Response Analysis for Oil-Immersed Transformers. IEEE Std C57.149-2012. [Online]. Available: http://standards.ieee.org/findstds/standard/C57.149-2012.html [June, 2014]

[2] Arduino. *Arduino Uno* [Online]. Available: http://arduino.cc/en/Main/ArduinoBoardUno [June, 2014].

[3] Arduino. *Arduino Yun Guide* [Online]. Available: http://arduino.cc/en/Guide/ArduinoYun [September, 2014].

[4] Analog Devices. (2004). "AD9850: CMOS, 125 MHz complete DDS synthesizer," [Online]. Available: http://www.analog.com/media/en/technical-documentation/data-sheets/AD9850.pdf [June, 2014].

[5] Texas Instruments. (2009). "OPA695: Ultra-wideband, current-feedback operational amplifier with disable," [Online]. Available: http://www.ti.com/lit/ds/symlink/opa695.pdf [September, 2014]

[6] Hittite Microwave Corporation. "HMC659: Gaas phemt mmic power amplifier, dc - 15 ghz," [Online]. Available: http://www.hittite.com/content/documents/data_sheet/hmc659.pdf [September, 2014]

[7] Texas Istruments (October, 2008), "THS3091: High-voltage, low-distortion, current-feedback operational amplifiers datasheet," [Online]. Available: http://www.ti.com/lit/ds/symlink/ths3091.pdf [October, 2014]

[8] Analog Devices Inc. (2002). "AD8302: Lf-2.7 GHz RF/IF gain and phase detector datasheet." [Online]. Available: http://www.analog.com/media/cn/technical-documentation/data-sheets/AD8302.pdf [July, 2014]

[9] Analog Devices. (2005). "AN-691: Operation of rf detector products at low frequency," [Online]. Available: http://www.analog.com/media/en/technical-documentation/application-notes/AN-691.pdf [July, 2014.]

[10] Processing. “Environment (ide) ”. [Online]. Available: https://processing.org/reference/environment/ [July, 2014]

[11] Plotly. “Real-time graphing and data logging,” [Online]. Available: https://github.com/plotly/arduino-api [September, 2014]

[12] Arduino. “analogread().” [Online]. Available: http://arduino.cc/en/Reference/analogRead [September, 2014].

[13] J. G. Carpio, “Grafica 2d plotting library for processing ide.” [Online]. Available: https://github.com/jagracar/grafica [October, 2014].

Appendix A

Project Budget

Please refer to Table A.I for the budget of the project. The initial budget for the project was $400.00 and was covered by the Dept. of Electrical and Computer Engineering. The total project cost is $343.69.

Table A.I: PROJECT BUDGET

System Element	**Component**	**Parts Number**	**Supplier**	**Quantity**	**Cost**
Input Source	Microcontroller	Arduino Yun	Digi-Key	1	$93.05
	VFO	AD9850 Module	eBay	1	$10.00
Amplifier	Power Amplifier	THS309X	eBay	1	$48.00
	DC Power Supply	+-15V DC	ECE Tech shop	1	$0.00
Comparator	Gain/Phase	THS309X	eBay	2	$31.55
Interface	Coding Environment	Processing	Processing.org	N/A	$0.00
Ciruit components	Wire Wrap	Wire Wrap Board	ECE Tech shop	1	$0.00
	Circuit Element	R,L,C	ECE Tech shop	N/A	$0.00
Simulation	NI MultiSim	NI MultiSim	ECE Dept	N/A	$0.00
				Total Taxes:	$39.54
				Total Project Cost:	$343.69

Appendix B

Project Code

B.1 Arduino Code

```
#include <Console.h>
#include <math.h>
#include <PlotlyYun.h>
#include <YunMessenger.h>
#define W_CLK 8 // Pin 8
-
connect to AD9850 module word load clock pin (CLK)
#define FQ_UD 9 // Pin 9
-
connect to freq update pin (FQ)
#define DA
TA 10 // Pin 10
-
connect to serial data load pin (DATA)
#define RESET 11 // Pin 11
-
connect to reset pin (RST).
#define pulseHigh(pin) {digitalWrite(pin, HIGH); digitalWrite(pin, LOW); }
plotly plotter1("6h4yap55qm");
plotly plotter2("mddd
w1o1fr");
double StartFeq = 10;
double EndFeq = 2e6;
int sample = 200;
double increment = (log10(EndFeq)
-
log10(StartFeq))/(sample
-
1);
float CurrentFeq;
double analoginput;
double power=log10(StartFeq);
float Gain;
float Phase;
int sign=1;
do
uble Vp;
double Vg;
// transfers a byte, a bit at a time, LSB first to the 9850 via serial DATA line
void tfr_byte(byte data)
{for (int i=0; i<8; i++, data>>=1) {
digitalWrite(DATA, data & 0x01);
pulseHigh(W_CLK); //after each bit sent, CLK is pulsed high}}
// frequency calc from datasheet page 8 = <sys clock> * <frequency tuning word>/2^32
void sendFrequency(double frequency) {
int32_t freq = frequency * 4294967295/125000000; // no
te 125 MHz clock on 9850
for (int b=0; b<4; b++, freq>>=8) {
tfr_byte(freq & 0xFF);}
tfr_byte(0x000); // Final control byte, all 0 for 9850 chip
pulseHigh(FQ_U
D);}
void setup() {
// configure arduino data pin
s for output
pinMode(FQ_UD, OUTPUT);
pinMode(W_CLK, OUTPUT);
pinMode(DATA, OUTPUT);
pinMode(RESET, OUTPUT);
pinMode(A0,INPUT);
pinMode(A1,INPUT);
Serial.begin(9600);
Bridge.begin();
Console.begin();
//while (!Console);
// wait for Consol
e port to connect.}
pulseHigh(RESET);
pulseHigh(W_CLK);
pulseHigh(FQ_UD); // this pulse enables serial mode
-
Datasheet page 12 figure 10}
void loop() {
CurrentFeq=pow(10,power);
//
Calculate
current frequency
sendFrequency(CurrentFeq);
//
S
end current frequency to AD9850
delay(3000);
Vg=analogRead(A0);
delay(100);
Vp=analogRead(A1);
Vg=Vg/188;
Vp=Vp/191;
//
Re
ad
Gain and phase value and convert them to voltage
convert(Vg,Vp);
//Call function convert, convert voltage to dB and d
egree
Display2();
//
Send data to processing
delay(500);
plotter1.plot(CurrentFeq,Gain);
delay(100);
plotter2.plot(CurrentFeq
,Phase);
//Send data to Plotly
if (CurrentFeq <= 2e6)
//if
current
frequency
is larger than 2MHz, Stop program
{power=power+increment;}
else{
while(1){
```

```
delay(100000);}}}
void convert(double Vg,double Vp){
/
convert
Vg and Vp [
voltage]
to dB and d
egree
Gain=
-
30+Vg/0.03;
Gain=pow(10,(Gain/20))/0.69;
Gain=20*log10(Gain);
if (Gain>=0){
Gain=0;}
if(sign==1){
Post(Vp);}
else{
Neg(Vp);}}
void Post(double Vp){
if(Vp>=1.79){
Phase=0;
sign=0;
Neg(Vp);}
else{
Phase=(1.8
-
Vp)/0.01;
sign=1;}}
void Neg(double Vp){
if(Vp>=1.79)
{Phase=0;}
```

B.2 Plotly Code

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
from YunMessenger import Console
import plotly.plotly as py
import json
import traceback
import datetime
import time
from pytz import timezone
import sys
import os
from plotly.graph_objs import *
# Initialize a YunMessenger console object
try:
c = Console.Console(log_filename="/root/Plotly.log")
except IOError as e:
print("ERROR: The log file directory that you specified does not exist")
raise e
# Load up plotly credentials
config_filename = '/root/config.json'
try:
with open(config_filename) as config_file:
config_string = config_file.read()
# Unicode song-and-dance to convert curly double quotes to straight single quotes
config_unicode = unicode(config_string, 'utf-8')
punctuation = { 0x2018:0x27, 0x2019:0x27, 0x201C:0x22, 0x201D:0x22 }
normalized_config_string = config_unicode.translate(punctuation).encode('ascii',
'ignore')
plotly_user_config = json.loads(normalized_config_string)
except IOError as e:
helpful_msg = "Please place a configuration file at {filename}"\
" with your plotly credentials and try again.\n"\
"Questions? Need some help? Support@plot.ly".format(filename=config_filename)
print(helpful_msg)
c.logger.error(helpful_msg)
raise e
except ValueError as e:
helpful_msg = "The JSON config file at {filename} could not be decoded, "\
"are you sure that it is in the right format?\n"\
"Questions? Support@plot.ly".format(filename=config_filename)
print(helpful_msg)
c.logger.error(helpful_msg)
raise e
# Extract the credentials and settings from the config file
username = plotly_user_config['plotly_username']
api_key = plotly_user_config['plotly_api_key']
stream_tokens = plotly_user_config['plotly_streaming_tokens']
file_option = plotly_user_config['file_option']
```

```
if 'show_this_many_points' in plotly_user_config:
maxpoints = plotly_user_config['show_this_many_points']
else:
maxpoints = 500
# Make the initializing plotly request
# Pass all stream tokens from the config file into this plot
py.sign_in(username, api_key)
trace0 = Scatter(
x=[],y=[],
name='Gain',
stream=Stream(
token='6h4yap55qm',
maxpoints=500))
trace1 = Scatter(
x=[],y=[],
name='Phase',
yaxis='y2',
stream=Stream(
token='mdddw1o1fr',
maxpoints=500))
data = Data([trace0,trace1])
layout = Layout(
title='Frequency Analyzer',
showlegend=True,
autosize=True,
width=1116,
height=589,
xaxis=XAxis(
title='Frequency',
range=[0, 6.1],
type='log',
autorange=True),
yaxis=YAxis(
title='Gain',
range=[0.5537603993344425, 1.1662396006655573],
type='linear',
autorange=True),
yaxis2=YAxis(
title='Phase',
range=[-2.5112701053799227, 39.93127010537992],
type='linear',
autorange=True,
anchor='x',
overlaying='y',
side='right'))
fig = Figure(data=data, layout=layout)
url = py.plot(fig,filename='Stream from Yun',auto_open=False,fileopt=file_option)
status = "Plot initialized at: {url}\nwith token(s): {tokens}"\
.format(url=url, tokens=', '.join(stream_tokens))
print(status)
c.logger.info(status)
class PlotlyHandler:
''' Create a message handler for streams
that writes to streams that are indexed by token
'''
def __init__(self, stream_tokens):
''' Store the stream tokens in an object
so that the message handler can access them
'''
self.streams = {}
for token in stream_tokens:
self.streams[token] = py.Stream(token)
self.streams[token].open()
```

B.3 System Interface Code

```
import grafica.*;
import java.util.Random;
//import de.bezier.data.*;
import processing.serial.*;
import java.io.BufferedWriter;
import java.io.FileWriter;
Serial myPort;
public GPlot plot1,plot2;
String []array; // hold gain, phase and frequency
float gain ;
float phase ;
float freq;
String outFilename = "Analog Data In.csv"; // CSV file name
public void setup() {
size(1200, 750);
println(Serial.list());
myPort = new Serial(this, "/dev/tty.usbmodemfa131", 9600);
```

```
//myPort.bufferUntil('\n');
plot1 = new GPlot(this);
plot1.setPos(0, 10); // Plot position
plot1.setDim(1000, 250);
plot1.getTitle().setText("Frequency Response Gain & Phase");
plot1.getXAxis().getAxisLabel().setText("Frequency");
plot1.getYAxis().getAxisLabel().setText("Gain");
plot2 = new GPlot(this);
plot2.setPos(0, 350);
plot2.setDim(1000, 250);
//plot2.getTitle().setText("Frequency Response");
plot2.getXAxis().getAxisLabel().setText("Frequency");
plot2.getYAxis().getAxisLabel().setText("Phase");}
public void draw() {
background(255);
String inString = myPort.readStringUntil('\n');
if (inString != null) {
inString = trim(inString); // trim off whitespaces.
array= split(inString,',');
if(array.length>2){
gain = float(array[0]);
phase = float(array[1]);
freq = float(array[2]);
println(gain+","+phase+","+freq);}}
if (freq>=1){
// Add a new point
GPoint lastPoint = plot1.getPointsRef().getLastPoint();
GPoint lastPoint2 = plot2.getPointsRef().getLastPoint();
//Gain
if (lastPoint == null) {
plot1.addPoint(freq, gain, "(" + str(freq) + " , " + str(gain) + ")");
//plot1.addPoint(freq, gain);}
else if (!lastPoint.isValid() || sq(lastPoint.getX() - freq) + sq(lastPoint.getY() + gain) >
0.01) {
plot1.addPoint(freq, gain, "(" + str(freq) + " , " + str(gain) + ")");
//plot1.addPoint(freq, gain);}
// Reset the points if the user pressed the space bar
if (keyPressed && key == ' ') {
plot1.setPoints(new GPointsArray());}
//Phase
if (lastPoint2 == null) {
plot2.addPoint(freq, phase, "(" + str(freq) + " , " + str(phase) + ")");}
else if (!lastPoint.isValid() || sq(lastPoint.getX() - freq) + sq(lastPoint.getY() + phase) >
0.01) {
plot2.addPoint(freq, phase, "(" + str(freq) + " , " + str(phase) + ")");}
// Reset the points if the user pressed the space bar
if (keyPressed && key == ' ') {
plot2.setPoints(new GPointsArray());}
//Save data to CSV file
File f = new File(dataPath(outFilename));
if(!f.exists()){}
try {
PrintWriter out = new PrintWriter(new BufferedWriter(new FileWriter(f, true))); // true
means append!
out.println(inString);
out.close(); // clean close allows to read / copy file
while processing runs
}catch (IOException e){
e.printStackTrace();}
// Draw the second plot
plot1.beginDraw();
plot1.drawBackground();
plot1.drawBox();
plot1.drawXAxis();
plot1.drawYAxis();
plot1.drawTitle();
plot1.drawGridLines(GPlot.BOTH);
plot1.drawLines();
plot1.drawPoints();
plot1.endDraw();
plot2.beginDraw();
plot2.drawBackground();
plot2.drawBox();
plot2.drawXAxis();
plot2.drawYAxis();
plot2.drawTitle();
plot2.drawGridLines(GPlot.BOTH);
plot2.drawLines();
plot2.drawPoints();
plot2.endDraw();}}
```

Appendix C

Test Results

C.1 Microcontroller and VFO Unit Test Results

Table C.I: Microcontroller and VFO Unit Test Results

ADC Result	Voltage	ADC Result	Voltage
3	0.014648	31	0.151367
5	0.024414	7	0.03418
5	0.024414	6	0.029297
5	0.024414	20	0.097656
5	0.024414	56	0.273438
6	0.029297	104	0.507813
7	0.03418	162	0.791016
7	0.03418	207	1.010742
9	0.043945	229	1.118164
9	0.043945	221	1.079102
10	0.048828	186	0.908203
12	0.058594	132	0.644531
24	0.117188	73	0.356445
63	0.307617	32	0.15625
112	0.546875	8	0.039063
169	0.825195	5	0.024414
212	1.035156	19	0.092773
230	1.123047	54	0.263672
218	1.064453	102	0.498047
179	0.874023	160	0.78125
124	0.605469	206	1.005859
66	0.322266	228	1.113281

27	0.131836	222	1.083984
7	0.03418	188	0.917969
7	0.03418	134	0.654297
23	0.112305	76	0.371094
60	0.292969	34	0.166016
109	0.532227	9	0.043945
166	0.810547	5	0.024414
210	1.025391	18	0.087891
229	1.118164	51	0.249023
219	1.069336	99	0.483398
181	0.883789	150	0.732422
126	0.615234	199	0.97168
69	0.336914	227	1.108398
29	0.141602	225	1.098633
7	0.03418	195	0.952148
6	0.029297	144	0.703125
22	0.107422	86	0.419922
58	0.283203	41	0.200195
107	0.522461	12	0.058594
165	0.805664	5	0.024414
209	1.020508	17	0.083008
229	1.118164	49	0.239258
220	1.074219	96	0.46875
183	0.893555	148	0.722656
129	0.629883	197	0.961914

C.2 System Test Results with Low-pass Filter

Table C.II: System test results with low-pass filter

	Gain Values		**Phase Values**	
Log10(Frequency)	Measured	Caculated	Measured	Caculated
1.079904468	1	0.999996371	-1	-0.154367629
1.106530854	1	0.999995897	-2	-0.164132388
1.133219457	1	0.999995361	-2	-0.174514824
1.159867847	1	0.999994756	-2	-0.185554008
1.186391216	1	0.999994072	-2	-0.197291481

1.212986185	1	0.999993298	-2	-0.209771411
1.239799818	1	0.999992423	-2	-0.22304076
1.266466895	1	0.999991434	-2	-0.237149461
1.2929203	1	0.999990316	-2	-0.252150602
1.319730494	1	0.999989052	-2	-0.26810063
1.346352974	1	0.999987624	-2	-0.28505956
1.372912003	1	0.999986008	-2	-0.303091205
1.399500661	1	0.999984182	-2	-0.322263409
1.426185825	1	0.999982118	-2	-0.342648307
1.452859336	1	0.999979784	-2	-0.364322594
1.479431337	1	0.999977146	-2	-0.387367813
1.50609896	1	0.999974163	-2	-0.411870664
1.532754379	1	0.999970791	-2	-0.437923321
1.5594278	1	0.999966979	-2	-0.46562379
1.586024382	1	0.999962669	-2	-0.495076263
1.612677918	1	0.999957797	-2	-0.52639152
1.639287226	1	0.99995229	-2	-0.559687337
1.665956029	1	0.999946063	-2	-0.595088927
1.692582562	1	0.999939024	-2	-0.632729411
1.719248398	1	0.999931067	-2	-0.672750313
1.745855195	1	0.999922071	-2	-0.715302092
1.772541733	1	0.999911902	-2	-0.760544695
1.799133693	1	0.999900405	-2	-0.808648161
1.825815445	1	0.999887409	-2	-0.859793249
1.852418993	1	0.999872717	-2	-0.91417211
1.879038505	1	0.999856108	-2	-0.971988997
1.905687968	1	0.999837332	-2	-1.033461025
1.932321532	1	0.999816108	-2	-1.098818966
1.958993665	1	0.999792114	-2	-1.168308105
1.985606083	1	0.999764992	-2	-1.242189132
2.01224652	1	0.999734332	-2	-1.320739104
2.038898212	1	0.999699674	-2	-1.404252447
2.065542371	1	0.999660497	-2	-1.493042027
2.092158996	1	0.999616212	-2	-1.58744028
2.118826663	1	0.999566154	-2	-1.687800403
2.145445036	1	0.999509572	-2	-1.794497615
2.172077257	1	0.999445617	-3	-1.907930486
2.19873954	1	0.99937333	-3	-2.028522333
2.22536098	1	0.999291627	-3	-2.156722697
2.252003021	1	0.999199285	-3	-2.293008889

2.278639298	1	0.999094921	-4	-2.437887611
2.305286865	1	0.998976976	-4	-2.591896653
2.331912949	1	0.998843687	-4	-2.755606671
2.358562977	1	0.998693066	-4	-2.929623031
2.385195018	1	0.998522869	-5	-3.114587727
2.411838481	1	0.998330564	-5	-3.311181368
2.438463235	1	0.998113294	-6	-3.520125218
2.465115053	1	0.997867837	-6	-3.742183289
2.491753783	1	0.997590562	-7	-3.97816448
2.518382316	1	0.997277377	-7	-4.228924722
2.545022443	1	0.996923672	-8	-4.49536914
2.571662256	1	0.996524256	-9	-4.778454184
2.598297954	1	0.996073287	-9	-5.079189706
2.624941805	1	0.995564197	-9	-5.398640941
2.651578426	1	0.994989601	-9	-5.737930347
2.67821782	1	0.994341209	-10	-6.098239242
2.704853745	1	0.993609717	-10	-6.480809168
2.731492065	1	0.992784694	-11	-6.8869429
2.758131883	1	0.991854464	-11	-7.318004995
2.784767024	1	0.990805968	-11	-7.775421765
2.811407422	1	0.989624621	-12	-8.260680535
2.838042695	1	0.988294167	-13	-8.775328014
2.864683104	1	0.986796511	-13	-9.32096761
2.891319977	1	0.985111561	-14	-9.899255453
2.917962145	1	0.983217054	-15	-10.51189489
2.944596166	1	0.981088393	-15	-11.16062921
2.971234088	1	0.978698479	-16	-11.84723219
2.997875455	1	0.976017566	-17	-12.57349634
3.024514401	1	0.973013125	-18	-13.34121828
3.051152522	1	0.969649746	-18	-14.15218102
3.077789275	1	0.96588908	-19	-15.00813281
3.104429065	1	0.961689835	-20	-15.91076214
3.131066625	1	0.957007853	-21	-16.86166872
3.15770452	0.9623506	0.951796275	-22	-17.86233014
3.184342152	0.9623506	0.946005821	-23	-18.91406427
3.21098163	0.9623506	0.939585203	-24	-20.01798734
3.23761913	0.9623506	0.932481704	-26	-21.17496814
3.264258072	0.9623506	0.92464192	-27	-22.38557897
3.290895665	0.9623506	0.916012704	-29	-23.65004411
3.317534761	0.9623506	0.906542303	-30	-24.96818735

3.344172123	0.9623506	0.896181694	-31	-26.33938012
3.370810906	0.9623506	0.8848861	-33	-27.76249251
3.397448109	0.9261187	0.87261666	-34	-29.23584966
3.424087425	0.9261187	0.859342192	-36	-30.75719622
3.450724724	0.9261187	0.845040996	-37	-32.32367204
3.477362945	0.8912509	0.829702588	-39	-33.9318019
3.504001653	0.8912509	0.813329286	-40	-35.57750176
3.530640175	0.8912509	0.795937521	-42	-37.25610388
3.557278566	0.8576959	0.77755877	-44	-38.96240172
3.58391709	0.8576959	0.758240011	-45	-40.6907148
3.61055477	0.8254042	0.738043628	-47	-42.43497247
3.637192426	0.7943282	0.717046714	-49	-44.18881389
3.663830912	0.7943282	0.695339787	-50	-45.94570076
3.690469865	0.7644223	0.673024944	-52	-47.69903785
3.717107719	0.7356423	0.65021355	-53	-49.44229627
3.743746446	0.7356423	0.627023578	-55	-51.16913356
3.770384341	0.7079458	0.60357675	-56	-52.87350558
3.797022965	0.6812921	0.579995633	-58	-54.54976543
3.823660688	0.6556418	0.556400836	-59	-56.19274578
3.850299396	0.6309573	0.532908453	-61	-57.79782225
3.876937411	0.6072022	0.50962785	-62	-59.36095683
3.903575582	0.5843414	0.486659866	-63	-60.87872138
3.930214233	0.5411695	0.464095475	-64	-62.34830262
3.956852482	0.5207948	0.442014898	-66	-63.76749071
3.983490521	0.5011872	0.420487152	-67	-65.13465387
4.010129004	0.4823178	0.399569985	-68	-66.44870223
4.036767384	0.4466836	0.379310137	-69	-67.70904365
4.063406173	0.4298662	0.359743865	-70	-68.91553444
4.090044558	0.413682	0.34089766	-71	-70.06842739
4.116683108	0.3831187	0.322789099	-72	-71.1683193
4.14332151	0.3686945	0.305427773	-73	-72.21609961
4.169959952	0.3548134	0.288816246	-74	-73.21290158
4.196598387	0.3285993	0.272951004	-75	-74.16005672
4.223236533	0.3162278	0.257823375	-75	-75.05905326
4.249875085	0.304322	0.243420383	-76	-75.91149883
4.276513733	0.2818383	0.229725542	-77	-76.71908748
4.30315241	0.2712273	0.216719572	-77	-77.48357094
4.329790906	0.2610157	0.204381027	-78	-78.20673398
4.356429544	0.2417315	0.192686858	-78	-78.89037348
4.383067808	0.2326305	0.181612893	-79	-79.53628099

4.409706285	0.2238721	0.171134245	-79	-80.14622835
4.436344917	0.2073322	0.161225664	-80	-80.72195608
4.462983601	0.1995262	0.151861821	-80	-81.26516415
4.489622158	0.184785	0.143017547	-81	-81.77750484
4.516260432	0.1778279	0.13466803	-81	-82.26057731
4.542898965	0.1711328	0.126788958	-81	-82.71592383
4.569537529	0.1584893	0.119356648	-82	-83.14502707
4.596176121	0.1525223	0.11234813	-82	-83.54930865
4.622814675	0.1467799	0.105741219	-82	-83.93012838
4.649452959	0.1359356	0.099514555	-82	-84.28878432
4.67609158	0.1308177	0.093647637	-82	-84.62651336
4.702730131	0.1258925	0.088120833	-83	-84.94449226
4.729368682	0.1165914	0.082915392	-83	-85.24383903
4.756007333	0.1122018	0.078013432	-83	-85.52561459
4.782645877	0.1039122	0.073397932	-83	-85.79082465
4.809284103	0.1	0.06905271	-84	-86.04042164
4.835922493	0.0926119	0.064962407	-84	-86.27530684
4.862561106	0.0891251	0.061112456	-84	-86.49633251
4.889199029	0.0857696	0.057489058	-84	-86.70430403
4.915837603	0.0794328	0.054079151	-84	-86.89998206
4.942476193	0.0764422	0.050870383	-84	-87.08408469
4.969114784	0.0735642	0.047851075	-84	-87.2572895
4.995753359	0.0681292	0.045010197	-84	-87.42023559
5.022391952	0.0681292	0.042337335	-84	-87.57352561
5.049030499	0.0630957	0.03982266	-84	-87.71772765
5.075669058	0.0630957	0.0374569	-84	-87.85337703
5.102307653	0.0607202	0.035231312	-84	-87.98097817
5.128946219	0.0584341	0.033137653	-84	-88.10100617
5.15558479	0.0584341	0.031168154	-84	-88.21390853
5.182222801	0.0562341	0.029315496	-84	-88.32010662
5.208861392	0.054117	0.027572783	-84	-88.41999716
5.235499939	0.054117	0.02593352	-84	-88.51395364
5.262138546	0.054117	0.024391592	-83	-88.60232763
5.288777125	0.054117	0.022941239	-83	-88.68545002
5.31541571	0.054117	0.02157704	-83	-88.76363222
5.342054322	0.054117	0.020293892	-83	-88.8371673
5.368692849	0.054117	0.019086992	-83	-88.90633103
5.395331444	0.0520795	0.017951818	-83	-88.97138288
5.421969995	0.0520795	0.016884116	-83	-89.03256698
5.4486086	0.0520795	0.015879882	-83	-89.09011305

5.475246578	0.0520795	0.01493535	-83	-89.14423718
5.501885184	0.0520795	0.014046975	-83	-89.19514268
5.528523782	0.0520795	0.013211422	-83	-89.24302079
5.555161765	0.0520795	0.012425554	-83	-89.28805143
5.58180033	0.0520795	0.011686418	-83	-89.33040384
5.608438909	0.0520795	0.010991239	-82	-89.37023723
5.635077502	0.0520795	0.010337404	-83	-89.40770135
5.661716099	0.0520795	0.009722456	-83	-89.44293706
5.688354655	0.0520795	0.009144083	-83	-89.47607686
5.714993237	0.0520795	0.008600111	-82	-89.50724538
5.741631832	0.0520795	0.008088495	-83	-89.53655984
5.768270371	0.0520795	0.007607311	-83	-89.5641305
5.794908976	0.0520795	0.00715475	-83	-89.59006107
5.821546974	0.054117	0.006729109	-83	-89.6144491
5.848185575	0.0520795	0.006328787	-82	-89.63738633
5.874824109	0.0520795	0.005952279	-82	-89.65895904
5.901462681	0.0520795	0.005598169	-82	-89.67924841
5.928101303	0.0520795	0.005265124	-82	-89.69833077
5.954739854	0.0520795	0.004951891	-82	-89.71627791
5.981378397	0.0520795	0.004657292	-82	-89.73315737
6.008017021	0.0520795	0.004380219	-82	-89.74903264
6.034655572	0.0520795	0.004119629	-82	-89.76396348
6.061294159	0.0520795	0.003874542	-82	-89.77800606
6.08793272	0.0520795	0.003644035	-83	-89.79121323
6.114570745	0.0520795	0.003427242	-82	-89.80363467
6.141209315	0.0520795	0.003223345	-82	-89.81531714
6.167847913	0.0520795	0.003031579	-82	-89.82630459
6.19448593	0.0520795	0.002851221	-82	-89.83663836
6.221124481	0.054117	0.002681593	-82	-89.84635736
6.247763061	0.0520795	0.002522057	-82	-89.85549815
6.274401609	0.0520795	0.002372012	-82	-89.86409512
6.301040228	0.054117	0.002230894	-82	-89.87218064

C.3 System Test Results with Band-pass Filter

Table C.III: System Test Results with Band-pass Filter

	Gain Values		Phase Values	
Log10(Frequency)	Measured	Calculated	Measured	Calculated
1	0.170412	0.002344	84.88	89.86567
1.03	0.170412	0.002493	84.88	89.85717
1.05	0.16125	0.00265	85.37	89.84814
1.08	0.164248	0.002818	85.37	89.83853
1.11	0.167302	0.002996	84.39	89.82832
1.13	0.167302	0.003186	84.88	89.81746
1.16	0.170412	0.003387	84.39	89.80591
1.19	0.167302	0.003602	84.88	89.79363
1.21	0.17378	0.00383	84.88	89.78058
1.24	0.167302	0.004072	84.88	89.7667
1.27	0.17378	0.004329	84.88	89.75194
1.29	0.164248	0.004603	84.88	89.73625
1.32	0.170412	0.004895	84.39	89.71956
1.35	0.164248	0.005204	84.39	89.70182
1.37	0.170412	0.005533	84.88	89.68296
1.4	0.170412	0.005883	84.88	89.6629
1.43	0.170412	0.006256	84.39	89.64158
1.45	0.170412	0.006651	84.39	89.61891
1.48	0.167302	0.007072	84.39	89.5948
1.51	0.170412	0.007519	84.39	89.56916
1.53	0.164248	0.007995	84.39	89.54191
1.56	0.167302	0.008501	84.39	89.51293
1.59	0.170412	0.009039	84.39	89.48211
1.61	0.167302	0.009611	84.39	89.44935
1.64	0.167302	0.010219	84.39	89.41451
1.67	0.167302	0.010865	84.39	89.37746
1.69	0.167302	0.011552	84.39	89.33808
1.72	0.167302	0.012283	84.88	89.2962
1.75	0.167302	0.013061	84.39	89.25166
1.77	0.167302	0.013887	84.39	89.20431
1.8	0.167302	0.014766	83.9	89.15396
1.83	0.167302	0.0157	84.39	89.10042
1.85	0.167302	0.016693	84.88	89.04349
1.88	0.167302	0.01775	84.39	88.98296

1.91	0.167302	0.018873	84.39	88.91859
1.93	0.167302	0.020068	84.39	88.85013
1.96	0.167302	0.021338	84.39	88.77735
1.99	0.167302	0.022688	84.88	88.69994
2.01	0.167302	0.024125	83.9	88.61763
2.04	0.167302	0.025652	84.39	88.53009
2.07	0.170412	0.027276	84.39	88.437
2.09	0.167302	0.029003	84.39	88.338
2.12	0.170412	0.03084	84.39	88.23271
2.15	0.167302	0.032794	84.39	88.12072
2.17	0.167302	0.034871	84.39	88.00162
2.2	0.167302	0.037081	84.39	87.87493
Gain	Phase	difference		
2.23	0.167302	0.039431	84.39	87.74018
2.25	0.167302	0.041931	84.39	87.59683
2.28	0.167302	0.04459	84.39	87.44434
2.31	0.167302	0.047419	83.9	87.2821
2.33	0.167302	0.050428	84.39	87.10948
2.36	0.167302	0.053629	84.39	86.92581
2.39	0.170412	0.057035	84.39	86.73035
2.41	0.167302	0.06066	83.9	86.52232
2.44	0.167302	0.064517	84.39	86.30089
2.47	0.167302	0.068622	83.9	86.06516
2.49	0.170412	0.072992	83.9	85.81417
2.52	0.167302	0.077643	83.9	85.54688
2.55	0.167302	0.082597	83.9	85.26217
2.57	0.167302	0.087871	83.9	84.95884
2.6	0.170412	0.09349	83.9	84.63558
2.62	0.167302	0.099476	83.9	84.29099
2.65	0.170412	0.105856	83.41	83.92353
2.68	0.170412	0.112656	83.41	83.53154
2.7	0.170412	0.119908	83.41	83.1132
2.73	0.170412	0.127644	83.41	82.66654
2.76	0.170412	0.1359	83.41	82.18936
2.78	0.170412	0.144714	82.93	81.67929
2.81	0.170412	0.15413	82.93	81.13368
2.84	0.17378	0.164194	82.44	80.54961
2.86	0.17378	0.174957	82.93	79.92382
2.89	0.17378	0.186478	82.44	79.25268
2.92	0.17378	0.198818	82.44	78.53214

2.94	0.177011	0.212048	81.95	77.7576
2.97	0.180302	0.226245	81.95	76.92388
3	0.180302	0.241497	81.46	76.0251
3.02	0.180302	0.2579	80.98	75.05452
3.05	0.183654	0.275563	80.49	74.00444
3.08	0.187284	0.294608	80.49	72.86595
3.1	0.190766	0.315173	79.51	71.62875
3.13	0.194312	0.33741	79.02	70.28083
3.16	0.197925	0.361492	79.02	68.80817
3.18	0.205589	0.387607	78.05	67.19433
3.21	0.21355	0.415964	77.56	65.41993
3.24	0.21752	0.446789	76.59	63.46217
3.26	0.225684	0.480313	75.61	61.29413
3.29	0.238781	0.516771	74.15	58.88412
3.32	0.252639	0.556368	72.2	56.19504
3.34	0.27227	0.599248	70.24	53.18391
3.37	0.298882	0.645434	67.8	49.8018
3.4	0.33458	0.694725	64.88	45.99472
3.42	0.381505	0.746567	61.95	41.70614
3.45	0.451336	0.799872	59.02	36.88214
3.48	0.524204	0.852818	54.15	31.48046
3.5	0.620155	0.902705	47.32	25.48405
3.53	0.720278	0.945983	38.05	18.91813
3.56	0.837529	0.978631	24.39	11.86614
3.58	0.91939	0.996949	6.34	4.476697
3.61	0.936483	0.998586	0	-3.04734
3.64	0.8531	0.983321	-22.93	-10.4793
3.66	0.762079	0.953143	-38.05	-17.6089
3.69	0.644169	0.911577	-48.78	-24.2759
3.72	0.544503	0.862653	-56.1	-30.3843
3.74	0.468274	0.810053	-61.46	-35.8989
3.77	0.395822	0.75665	-65.37	-40.8302
3.8	0.353997	0.704426	-68.29	-45.2168
3.82	0.310456	0.654592	-71.22	-49.1112
3.85	0.282488	0.607792	-73.66	-52.57
3.88	0.26212	0.564278	-75.61	-55.6478
3.9	0.24322	0.524064	-77.56	-58.3947
3.93	0.230144	0.487024	-79.02	-60.8549
3.96	0.221564	0.452958	-80	-63.0664
3.98	0.21752	0.421637	-80.49	-65.062

4.01	0.205589	0.392828	-81.46	-66.8694
4.04	0.197925	0.366302	-82.44	-68.5123
4.06	0.197925	0.341849	-82.93	-70.0105
4.09	0.190766	0.319274	-83.41	-71.381
4.12	0.187284	0.298403	-83.9	-72.6383
4.14	0.187284	0.279079	-83.9	-73.7947
4.17	0.183654	0.261163	-84.39	-74.8609
4.2	0.180302	0.244529	-84.88	-75.846
4.22	0.180302	0.229066	-85.37	-76.7579
4.25	0.177011	0.214675	-85.37	-77.6036
4.28	0.17378	0.201267	-85.37	-78.3889
4.3	0.177011	0.188763	-85.85	-79.1194
4.33	0.17378	0.177091	-85.85	-79.7996
4.36	0.170412	0.166188	-85.85	-80.4338
4.38	0.170412	0.155995	-86.34	-81.0255
4.41	0.170412	0.146459	-86.34	-81.5782
4.44	0.17378	0.137534	-86.34	-82.0948
4.46	0.170412	0.129175	-86.34	-82.5781
4.49	0.170412	0.121343	-86.83	-83.0304
4.52	0.170412	0.114002	-86.83	-83.454
4.54	0.170412	0.107118	-86.83	-83.8508
4.57	0.170412	0.10066	-86.83	-84.2228
4.6	0.170412	0.094601	-87.32	-84.5716
4.62	0.170412	0.088914	-87.32	-84.8989
4.65	0.167302	0.083576	-87.32	-85.2059
4.68	0.167302	0.078563	-87.32	-85.494
4.7	0.167302	0.073855	-87.32	-85.7646
4.73	0.167302	0.069433	-87.32	-86.0186
4.76	0.170412	0.065279	-87.32	-86.2571
4.78	0.167302	0.061376	-87.32	-86.4812
4.81	0.170412	0.057708	-87.32	-86.6917
4.84	0.167302	0.054262	-87.8	-86.8895
4.86	0.167302	0.051022	-87.32	-87.0754
4.89	0.167302	0.047977	-87.8	-87.25
4.92	0.167302	0.045115	-87.8	-87.4142
4.94	0.167302	0.042425	-87.8	-87.5685
4.97	0.164248	0.039895	-88.29	-87.7136
5	0.167302	0.037517	-88.29	-87.8499
5.02	0.167302	0.035282	-87.8	-87.9781
5.05	0.170412	0.03318	-87.8	-88.0986

5.08	0.167302	0.031203	-87.32	-88.2119
5.1	0.170412	0.029344	-87.32	-88.3184
5.13	0.167302	0.027597	-87.32	-88.4186
5.16	0.170412	0.025954	-87.32	-88.5128
5.18	0.170412	0.024408	-86.83	-88.6014
5.21	0.170412	0.022955	-87.32	-88.6847
5.24	0.167302	0.021589	-87.32	-88.763
5.26	0.170412	0.020304	-87.32	-88.8366
5.29	0.167302	0.019095	-87.32	-88.9059
5.32	0.170412	0.017958	-87.32	-88.971
5.34	0.170412	0.01689	-87.32	-89.0322
5.37	0.167302	0.015884	-87.32	-89.0898
5.4	0.167302	0.014939	-87.32	-89.144
5.42	0.170412	0.01405	-87.32	-89.195
5.45	0.170412	0.013214	-87.32	-89.2429
5.48	0.167302	0.012428	-87.32	-89.2879
5.5	0.170412	0.011688	-86.83	-89.3303
5.53	0.167302	0.010993	-87.32	-89.3701
5.56	0.167302	0.010339	-87.32	-89.4076
5.58	0.170412	0.009724	-87.32	-89.4429
5.61	0.170412	0.009145	-87.32	-89.476
5.64	0.167302	0.008601	-87.32	-89.5072
5.66	0.167302	0.008089	-86.83	-89.5365
5.69	0.167302	0.007608	-87.32	-89.5641
5.71	0.167302	0.007155	-87.32	-89.59
5.74	0.167302	0.006729	-87.32	-89.6144
5.77	0.170412	0.006329	-87.32	-89.6374
5.79	0.170412	0.005953	-87.32	-89.6589
5.82	0.170412	0.005598	-87.32	-89.6792
5.85	0.170412	0.005265	-86.83	-89.6983
5.87	0.170412	0.004952	-87.32	-89.7163
5.9	0.167302	0.004657	-87.32	-89.7331
5.93	0.167302	0.00438	-87.32	-89.749
5.95	0.167302	0.00412	-87.32	-89.764
5.98	0.170412	0.003875	-87.32	-89.778
6.01	0.170412	0.003644	-87.32	-89.7912
6.03	0.167302	0.003427	-87.32	-89.8036
6.06	0.170412	0.003223	-87.32	-89.8153
6.09	0.170412	0.003032	-87.32	-89.8263
6.11	0.170412	0.002851	-87.32	-89.8366

6.14	0.170412	0.002682	-87.32	-89.8464
6.17	0.170412	0.002522	-87.32	-89.8555
6.19	0.167302	0.002372	-87.32	-89.8641
6.22	0.170412	0.002231	-87.32	-89.8722
6.25	0.167302	0.002098	-87.32	-89.8798
6.27	0.170412	0.001973	-87.32	-89.8869
6.3	0.170412	0.001856	-87.32	-89.8937