



ECE 493 Senior Advance Design Project

Affordable USB Oscilloscope Final Project Report

Team members: Phu Duong
Duy Luong
Jasem A. J. M. Alsaei
Bao Phan
Giang Nguyen
Thu Viet Minh Nguyen

Faculty Advisor: Dr. Jens-Peter Kaps

ECE 493

Date of Submission: May 5, 2023

Table of Contents

1. Executive Summary	5
2. Problem Statement.....	5
2.1 Motivation and Identification of Need	5
2.2. Market Review	6
3. Approach	7
3.1 Problem Analysis	7
3.2 Our Approach.....	7
3.2.1 Overview	7
3.2.2 Approach Hardware	7
3.2.3 Approach Microprocessor	7
3.2.4 Approach Graphical User Interface	8
3.3 Alternative Approaches	8
3.3.1 Plugin Board.....	8
3.3.2 PyQtGraph.....	9
3.3.3 Alternative MCUs	9
3.4 System Requirement Specifications	10
3.4.1 Mission Requirements.....	10
3.4.2 Operational Requirements.....	10
4. System Decomposition & Architecture	11
4.1 Level Zero Decomposition	11
4.2 Level One Decomposition	12
4.3 Level Two Decomposition.....	12
5. Background Knowledge.....	14
5.1 Analog Front-End.....	14
5.1.1 Attenuator	14
5.1.2 AC/DC coupling	15
5.1.3 Voltage Gain Amplifier.....	15
5.1.4 DC Offset using PWM Method	16
5.1.5 Unity Gain Buffer	17
5.1.6 RC Low Pass Filter	18
5.2 MCU	19

5.2.1 ADC Converters	19
5.2.2 DMA	19
5.2.3 STM32CubeIDE	19
5.2.4 HAL Driver	19
5.2.5 LL Driver.....	20
5.3 GUI.....	20
6. Detailed Design	21
6.1 Analog Front-End Schematics	21
6.1.1 Power Supply -5V Schematic.....	21
6.1.2 USB Soft Start Schematic	22
6.1.3 Attenuator Schematic	23
6.1.4 AC/ DC Coupling Schematic	25
6.1.5 Gain Amplifier Schematic.....	25
6.1.6 DC Offset using PWM, Buffer, and RC Low Pass Filter Schematic	27
6.1.7 Buffer and RC Low Pass Filter, and Voltage Protection Schematic	28
6.1.8 The Whole Design Schematic	29
6.2 PCB Analog Front-End	29
6.3 MCU Design	32
6.4 GUI Design	37
7. Experimentation and Success Evaluation	39
7.1 Experimentation of MCU	39
7.1.1 Experiment 1: USB	39
7.1.2 Experiment 2: GPIO.....	41
7.1.3 Experiment 3: USB continue.....	42
7.1.4 Experiment 4: MCU Testing with Front-end	43
7.1.5 Experiment 5: ADC Speed Testing.....	45
7.2 Experimentation of GUI	45
7.2.1 Experiment 1: Reading Data from The USB Port Testing.....	45
7.2.2 Experiment 2: Functions Testing.....	46
7.2.3 Experiment 3: Communicate to MCU Testing.....	46
7.3 Front-End/MCU/GUI Experimentation.....	47
7.3.1 The Whole Project Testing.....	47

7.3.2 Frequency Response for Front-End PCB	49
7.4 Project Success Evaluation	51
7.4.1 Overall Project Evaluation	51
7.4.2 Other Issues	51
8. Administrative Project Aspects	52
8.1 Project Progress.....	52
8.2 Project Challenges.....	52
8.3 Funds Spent.....	53
8.4 Man-Hours Devoted to The Project	53
8.5 Individual Team Member Contributions	54
9. Lessons Learned	54
9.1 Additional Knowledge and Skills Learned.....	54
9.2 Teaming Experience.....	55
9.2.1 Small Team.....	55
9.2.2 Team Communication.....	55
9.2.3 Time tracking.....	55
10. References	56
11. Appendix A: Proposal (ECE-492)	56
12. Appendix B: Design Document (ECE-492)	80
13. Appendix C: BOM for single channel Front End Design	130

1. Executive Summary

This project involves the design and construction of an affordable USB Oscilloscope device that meets study requirements of undergraduate Electrical and Computer Engineering (ECE) curriculum. The primary aim of the task is to design and construct an affordable USB Oscilloscope which will have a low cost even if the chip is in a shortage crisis condition. Secondly, the performance of the design must have a high accuracy. Finally, the design should have the same features as other oscilloscopes in the market such as AC/DC coupling, attenuator, amplifier, DC offset, buffer, low pass filter, and voltage protection. The device shall use the custom PCB design for the analog front end and microcontroller that have interface to the open source software (PyQtGraph) via USB connection to display and navigate the results.

Initially in ECE 492, this project will be divided into three small separate teams such as front end, microcontroller, and GUI. Each team member will have their own task to do research and break down the problems to understand clearly their assigned part from the project. After coming up with the sketches and designs, the front-end, microcontroller, and GUI will be tested individually or in combination with other teams in the project throughout the period of ECE 492. For the front-end part, the input going through the circuit must have the correct output as we calculated by formulas. Next, the microcontroller (MCU) must perform ADC fast enough and not miss data while transmitting to GUI. Finally, the GUI must receive/transmit data from/to MCU and perform live waveform generation. After the individual testing period, each part will be fixed and improved to be more efficient and meet the main goal. At the very end of ECE 492, three groups will come together to start coming up with a full custom PCB design for the real affordable USB Oscilloscope device with the full connection of three parts. Moving forward to ECE 493, this will be the period of building and testing the device as a whole.

2. Problem Statement

2.1 Motivation and Identification of Need

For many years, undergraduates of the Electrical and Computer Engineering department have been necessitated to purchase the USB Laboratory Instruments such as the Analog Discovery 2 (AD2) or Advanced Active Learning Module (ADALM 2000). These USB Laboratory Instruments help students to learn and conduct lab experiments in their own time for their convenience without depending too much on the open hours of the lab rooms on campus. Thanks to the convenience of these devices, during COVID-19 period, students were able to use the Analog Discovery 2 and Advanced Active Learning Module to continue their studying at home safely without any obstacles. The pandemic has resulted in a high demand for these devices, as students around the country are trying to get a hand on it for their study. The surge in demand has challenged manufacturers around the world to quickly adapt and respond. Unfortunately, this outbreak has resulted in a serious chip shortage, which has led to a serious soaring of price for these devices.

Currently, the AD2 will cost around \$399 and the ADALM2000 will cost around \$252.79, which not every student can afford this overwhelming price for these devices. To reduce the significant amount of money that students need to invest for the equipment, this project will develop a less expensive oscilloscope solution compared to the current market. The new affordable USB Oscilloscope will include a communication port by USB, 64GB RAM to transmit and receive data, 2 channels 12-bit ADC with 5Msps, at least 2Mhz clock speed, and enough general-purpose input and output to connect with the front end. This new device can measure the scope of AC/DC signals with an average accuracy and precision measurement in order to meet study requirements of undergraduate Electrical and Computer Engineering curriculum.

2.2. Market Review





Model	Device Picture	Bandwidth (MHz)	Sampling Rate (MSa/s)	Number of Analog Channels	Sampling Resolution (bit)	Price (\$)
Analog Discovery 2 (AD2)		30	100	2	14	\$399
Advanced Active Learning Module (ADALM2000)		10	100	2	12	\$252.79
Hantek 2D72 - Handheld Oscilloscope with Digital Multimeter		20	250	2	8	\$174
FNIRSI-5012H 2.4-inch Screen Digital Handheld Pocket Oscilloscope		100	500	1	8	\$76.99

Table 1: Current Open Source Alternative Oscilloscopes

3. Approach

3.1 Problem Analysis

In order to solve the problems that we described above; the designed system has to meet these requirements: the device hardware should be low-cost while still fulfilling the requirements for most laboratory exercises in the undergraduate ECE curriculum. The total part cost should not exceed \$50. This is a tough challenge for our team to overcome because some electronic devices like chips, microcontrollers, boards... are in short supply and more expensive than before. Moreover, we have to make sure this laboratory instrument can measure the scope of AC/DC signals with an average accuracy and precision measurement. Therefore, in order to accomplish these requirements, we will be very selective with our design and component choices.

3.2 Our Approach

3.2.1 Overview

To solve the problem above, we design a device, and it must overcome three challenges. First, it is hardware cost, it must be low cost even if the chip is in a shortage crisis condition. The second is performance, the design must have high accuracy. Finally, the design should have the same features as other oscilloscopes in the market, and the design has more features than other low-cost oscilloscopes.

3.2.2 Approach Hardware

To meet the project requirements, we must consider the capabilities and features of an oscilloscope, such as AC/DC coupling, attenuator, amplifier, DC offset, buffer, low pass filter, and voltage protection. The ADC of the microcontroller accepts input voltages in the range of 0 to 3.3 volts, with attenuators having two scales of 1.25 and 12.5 for voltage ranges of (0.1 - 4) V_{pp} and (4 - 40) V_{pp}, respectively. A one-channel analog switch will be used to change the attenuator scale, while a two-channel analog switch will be used for switching the amplifier with four scales of 1, 2.5, 5, and 10. We will use a switch to toggle between AC/DC coupling, and the buffer will serve as a current amplifier. To eliminate signal noise above 500 kHz, we will use a low pass filter, as the device can only measure frequencies up to 500 kHz. We will use a DC offset with PWM to adjust the signal to the right range of the ADC, which only receives positive voltages from 0 to 3.3 volts. Each circuit will be designed with components chosen based on calculation formulas and information from the component datasheets, with a focus on minimizing cost.

3.2.3 Approach Microprocessor

We have to consider what microprocessor is fit for our design. To get the right microprocessor, we must know what features are built in the microprocessor. For example, how much RAM does it have? How does a microcontroller transmit and receive data to or from a PC (USB/UART/Wi-Fi)? How many ADC bits and ADC channels does it have? What are the sampling rate and clock speed? The last consideration is the price and the availability of a microcontroller in the market.

After reviewing 3 microcontrollers, such as the MSP430G2553 launchpad, STM32F302R8T6 Nucleo board, and STM32F303RET6 Nucleo board. We know that the first two do not meet

requirements of the project. The RAM must be more than 64kB to store ADC data, and the microcontroller has 2 input channels. The first one, MSP430G2553 launchpad, has 512 RAM and 10-bit ADC. The second one, STM32F302R8T6 Nucleo board, has 16k RAM, 12-bit ADC, and 1 ADC channel. The last one is the STM32F303RET6 Nucleo board. It has 80k RAM, 4 ADC channels. It meets our project's requirements.

3.2.4 Approach Graphical User Interface

The GUI is the primary interface between the user and the overall system. It will allow the user to enter the system configuration command and display a waveform data. The questions: how does the board connect to the PC? What is the script language to create the GUI system? Which application framework do we use? How does the GUI communicate to the board?

The GUI section of this project is done using Python as its base. Serial data is collected using PySerial, processed using NumPy, and finally plotted using PyQtGraph with communication buttons that connect it to the microcontroller. PySerial will receive data from the microcontroller using USB into the serial port of the computer. The data collected from USB is processed into arrays and smoothed using NumPy. Finally, the processed data is sent to PyQtGraph to be plotted and show the signal output.

3.3 Alternative Approaches

3.3.1 Plugin Board

This alternative approach comes from the chip shortage. Since the STM32F303RE single chip is hard to get, instead of disordering the chip from the Nucleo board which has a chance of destroying the chip, another option is to design a plugin board that can connect directly to the Nucleo development board. This approach is similar to some of the Arduino shields that add additional functionality to the main board.

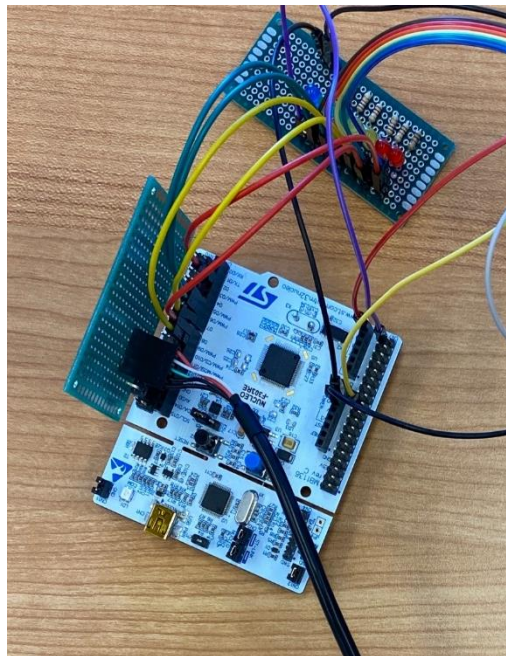


Figure 1: Arduino shield addon Board

This approach might introduce complexity to the overall design. It might also introduce more points of failure which makes it hard to figure out what's wrong.

3.3.2 PyQtGraph

Another approach that might make it in the end is a Python extension that makes it easy to create plots and has a graphical user interface called PyQtGraph.

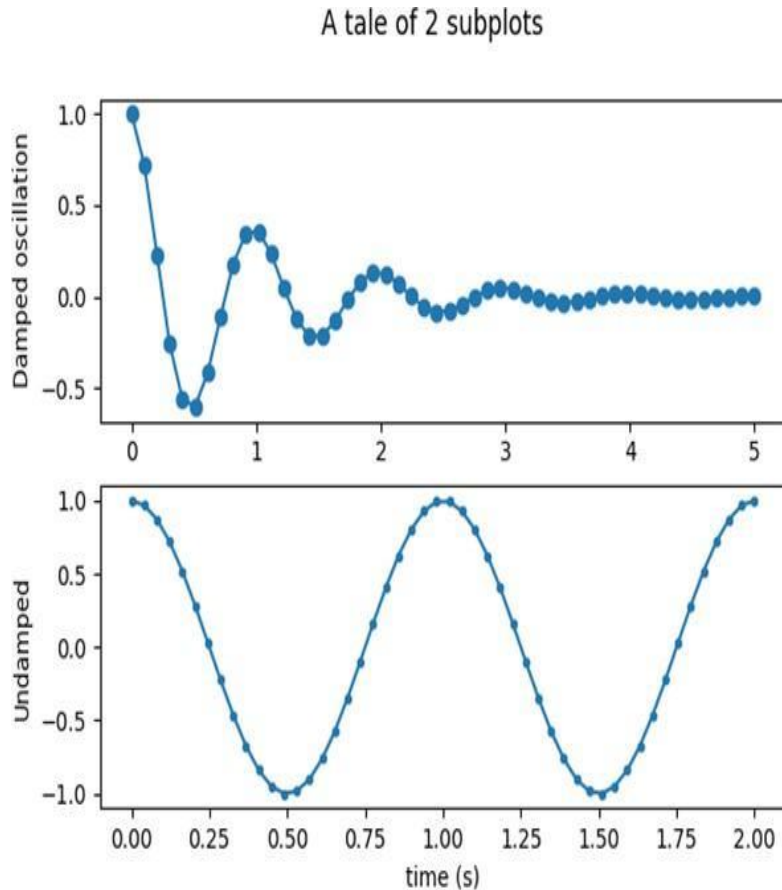


Figure 2: PyQtGraph signal plot example

PyQtGraph has an easy-to-use fast interface and has a simple backend. Scripts can be used to collect the signal input data from the MCU and plot it into a graph with user controls. However, PyQtGraph lacks the professional and familiar look of Waveforms Live, and needs to be created from scratch to turn it into a full-fledged.

3.3.3 Alternative MCUs

Currently, the top pick is the STM32F303RE MCU for its large flash/RAM capacity vs price. This project needs a large RAM capacity to store the signal data that is to be imported to the computer via USB. However, if the RE version were to not be available, the STM32F303RD can be used since it has the same RAM capacity but with reduced Flash capacity.

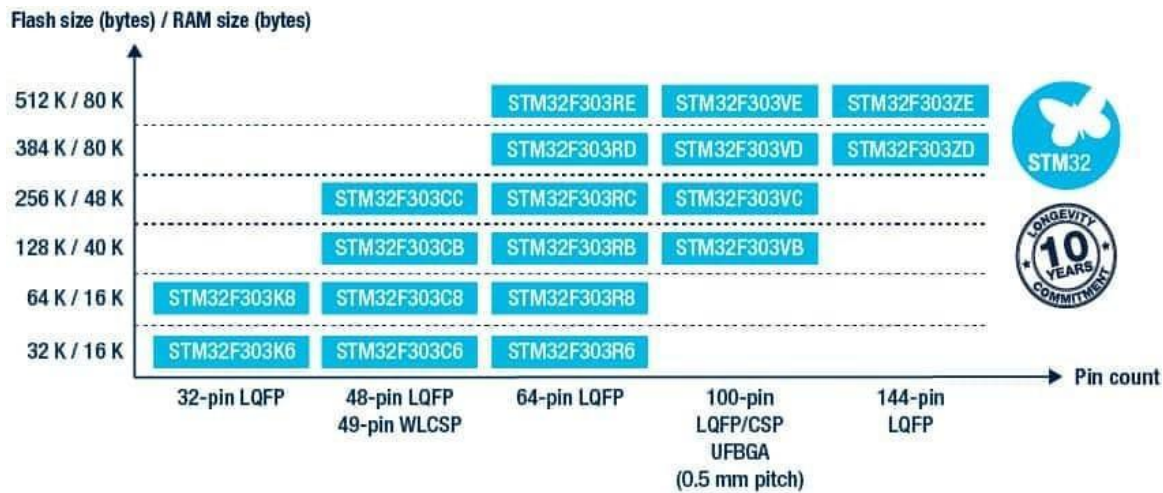


Figure 3: STM32F303 Series

3.4 System Requirement Specifications

3.4.1 Mission Requirements

This project involves the design and construction of an affordable USB Oscilloscope device that meets study requirements of undergraduate Electrical and Computer Engineering (ECE) curriculum. The primary aim of the task is to design and construct an affordable USB Oscilloscope which will have a low cost. Secondly, within an affordable price, any student can purchase and play around with the device. Finally, the design should have the same features as other oscilloscopes in the market such as AC/DC coupling, attenuator, amplifier, DC offset, buffer, low pass filter, and voltage protection. The device shall use the custom PCB design for the analog front end and microcontroller that have interface to the open source software via USB connection to display and navigate the results.

3.4.2 Operational Requirements

- Input/output requirements
 - The device shall have two analog input channels and the output will display the signal on the LCD via the open source software. The users are able to change the configuration or setting that they want with the output signal.

Oscilloscope

- Channels: 2
- Input impedance 1 M Ω
- 12 bits ADC
- Ram \geq 64KB
- Sample Rate 5 Msps
- Bandwidth 2 MHz
- Useful Max Frequency 500 kHz
- Voltage Range +/- 20V
- USB

- The device has the working range input from -20V to +20V

- The input impedance is 1 MOhm
- The signal bandwidth is 2 MHz
- External Interface Requirements
 - The device will use the power source/ communication from the USB
- Function requirements
 - The device will use the microcontroller with analog front end to analyze the input signal as the following specification
 - The device will have the trigger option (GPIO trigger/Level trigger)
 - The max frequency that device can measure is 500KHz
 - The device will have a sample rate 5 MS/s
 - The device will have a DC offset.
 - Memory for channels is 40KB, each channel is 20KB
- Software requirements
 - The software application can run on Linux, Window, OSX
 - It should run on any platform which supports the following packages: python, PyQtGraph, NumPy, C, C+, ...
 - Speed minimum is 10 fps
 - Easy to use and operation
- Technology and system-wide requirement
 - The analog front end with custom PCB design will be interface with microcontroller in order to process the input data
 - The microcontroller shall have at least 2 ADCs, RAM size is greater than 64K for fast respond
 - The device should be as small as possible
 - The device should be low-cost (less than \$50)
 - USB is using to data communication

4. System Decomposition & Architecture

4.1 Level Zero Decomposition

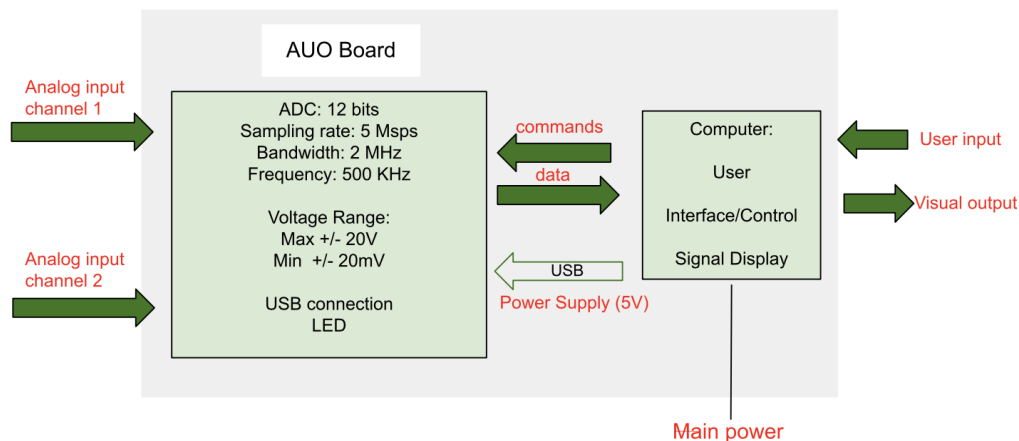


Figure 4: Level Zero Functional Architecture Block Diagram

The Level zero provides the overview of the top down oscilloscope design. The system is shown as in figure below which include 2 analog input channels, user command, DC power supply from USB. The output is the waveform signal which will be shown in the GUI.

4.2 Level One Decomposition

The design is in the detail as the figure below, signal will be conditioning by the Front End then the microcontroller which has an ADC function will convert/ process the analog signal to digital signal. By the communication through the USB, the data is transmitted to the GUI then the waveform will be plotted to visualize the analog signal on the monitor.

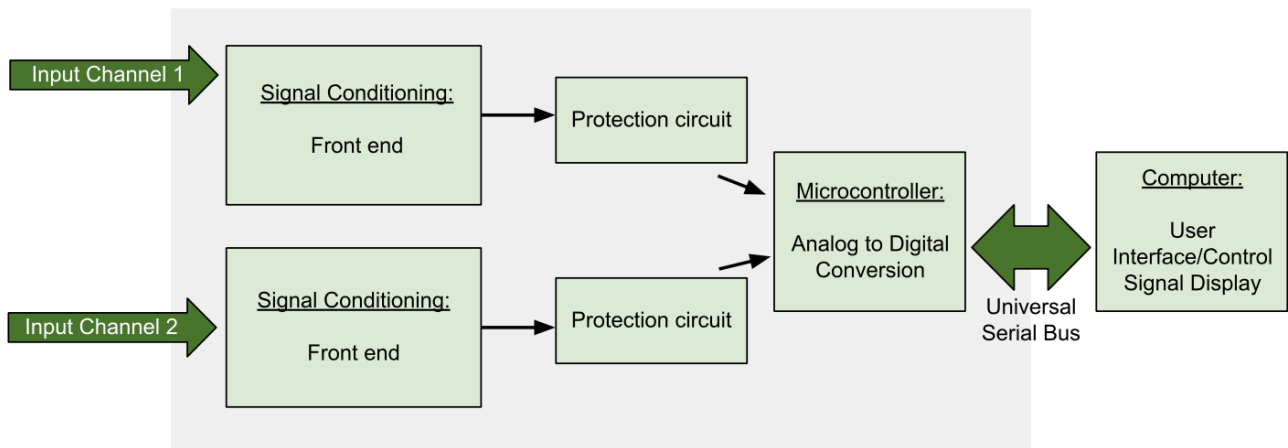


Figure 5: Level One Functional Architecture Block Diagram

4.3 Level Two Decomposition

Level 2: Front-End

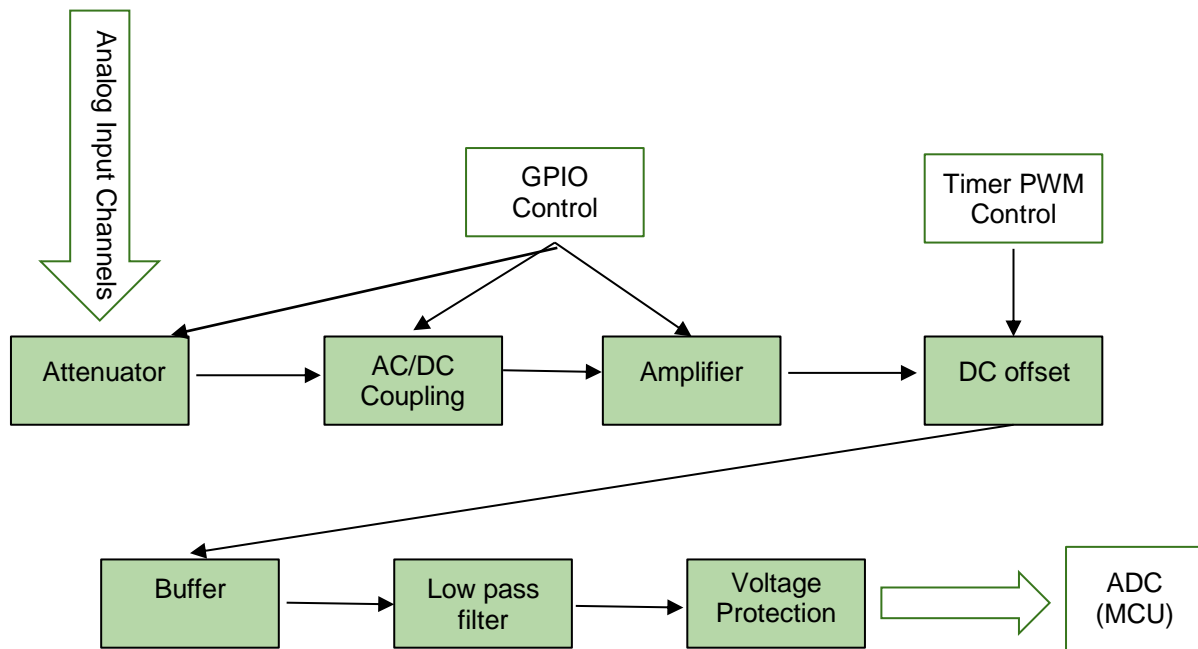


Figure 6: Level Two Functional Architecture Block Diagram - Analog Input Signal Preconditioning

The main hardware for the oscilloscope is the analog front end that is starting with the attenuator which is used to scale down the input signal. The DC/AC coupling function is used to block/unblock the DC offset that we want in the signal. The amplifier is used to scale up the signal to the range that the ADC can be read. Different scales will be controlled by analog switches by GPIO signal from the microcontroller. In addition, the signal will be offset above zero for the ADC is able to read. to strengthen the current, the buffer is added to provide the needed current to send to the ADC. The low pass filter will help to filter out the noise signal that is not needed. and finally, the voltage protection is used to protect the unwanted high voltage that suddenly appear to damage the microcontroller.

Level 2: MCU

The MCU will receive the signal from the front-end and transfer it to the computer (GUI). The MCU includes GPIO, Timer PWM, ADC, DMA, and the USB. The GPIO will control the AC/DC coupling and amplifier of the front-end. The GPIO will choose specific options in AC/DC coupling and amplifier circuits to process signals. Also, the timer PWM will control the DC offset of the front-end by variable output voltage to shift signal up and down. The signal received from the front-end will go to the ADC, then move forward to DMA and then the USB. From the USB, there will be 2 feedbacks to the Timer PWM and the GPIO. The MCU will use USB to transfer/receive data to/from the GUI. In transferring, the MCU transfers signal under digital form to GUI. In receiving, the MCU receives a signal from the GUI to change the FE setup, adjust the signal before it goes through the MCU.

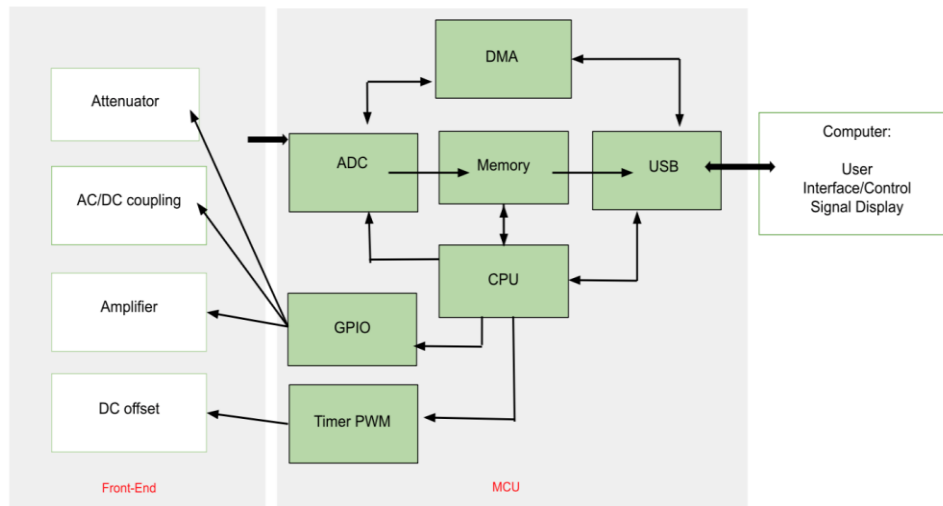


Figure 7: Level Two Functional Architecture Block Diagram - Microcontroller

Level 2: GUI

After the memory dump through USB happens, the data will be moved from the ADC into the main local machine. This data is interpreted by the computer as a hexadecimal number which needs to be further processed before being plotted into the live plot. The conversion from hexadecimal to a voltage number is needed because the cursor function of the plot will show whatever row value is shown which is why having the voltage value is correct. Furthermore, the signal will be capped between 0 to 3.3v because of the ADCs limitations, this means that the input signal will not be the same as the data out of

the ADC which is where further processing is needed to return the signal back to its original form before plotting.

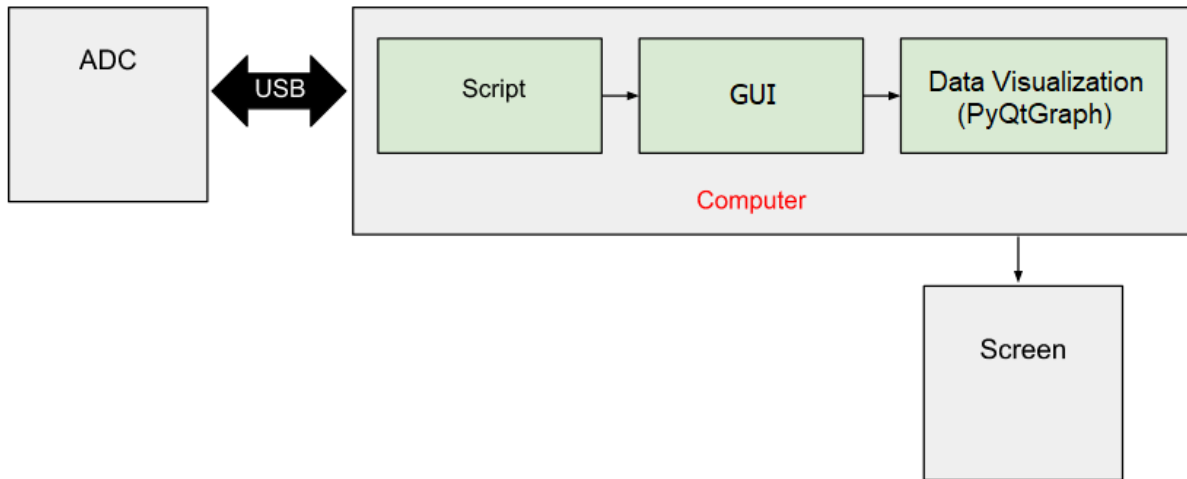


Figure 8: Level Two Functional Architecture Block Diagram - GUI

- User Interface

The GUI section of this project is done using Python as its base. Serial data is collected using PySerial, processed using NumPy, and finally plotted using PyQtGraph with communication buttons that connect it to the microcontroller. PySerial will receive data from the microcontroller using USB into the serial port of the computer. The data collected from USB is processed into arrays and smoothed using NumPy. Finally, the processed data is sent to PyQtGraph to be plotted and show the signal output.

5. Background Knowledge

5.1 Analog Front-End

The Analog Front End includes multiple sections in order to provide the correct input signal to the ADC of the MCU. There are attenuators, AC/DC coupling, gain amplifiers, low pass filter, and DC offset using PWM in our front-end design. The basic background knowledge and calculation used to select the hardware components are listed below.

5.1.1 Attenuator

The attenuator uses a simple resistor to do voltage divider. To prevent the capacitance effects at the high frequency, the capacitors are connected parallel to the resistor which needs to satisfy the equation below.

$$R_1 * C_2 = R_2 * C_1$$

$$V_2 = V_S \frac{R_2}{R_1 + R_2}$$

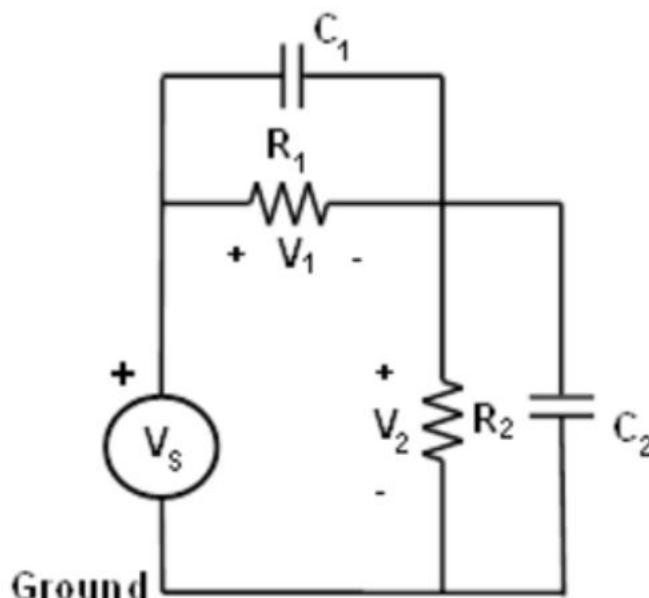


Figure 9: A Frequency Compensated Voltage Divider

5.1.2 AC/DC coupling

AC and DC stand for Alternating (Capacitive) Coupling and Direct Coupling, respectively. This parameter is critical since it affects the frequency content of the signals. The majority of signals are made up of both AC and DC components. The DC component is the 0 Hz component that functions as a time domain offset. For more detail, AC coupling eliminates DC offset by connecting a DC-blocking capacitor in series with the signal. AC coupling effectively rejects the signal's DC component, standardizing it to a mean of zero. On the other hand, DC (direct coupling) allows both alternating current and direct current signals to travel across a link. The DC component is a 0 Hz signal that functions as an offset for the AC component of the signal [2].

$$C = \frac{1}{2\pi f}$$

5.1.3 Voltage Gain Amplifier

An amplifier is a type of electronic device that may amplify the amplitude of a signal (a time-varying voltage or current). It is a two-port electrical circuit that uses electricity from a power source to enhance the amplitude of a signal supplied to its input terminals, resulting in a higher amplitude signal at its output. Gain is the ratio of output voltage to input voltage [3]. The figure below is the non-inverted op amp configuration and the gain will be calculated by.

$$A_v = 1 + \frac{R_f}{R_2}$$

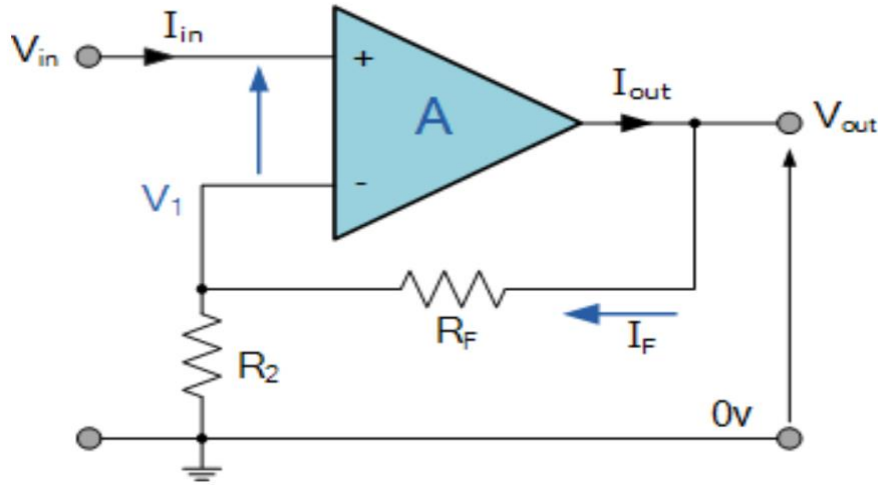


Figure 10: Voltage Gain Amplifier Circuit

5.1.4 DC Offset using PWM Method

DC offset is an average amplitude deviation from zero. When the signal goes out of the range ADC of the oscilloscope, the DC offset adds the Dc voltage to the incoming signal to move it in the range [1].



Figure 11: Range Signal by Analog Offset

For many Microcontrollers, the default output is pulse width modulation. It may be necessary to convert a PWM to a DC signal in order to communicate with an external circuit or device because that circuit or device may need a variable DC output.

$$\text{Desired DAC Voltage} = A * \text{duty cycle}$$

with A is the peak value of the square wave and duty cycle = (width / period) * 100

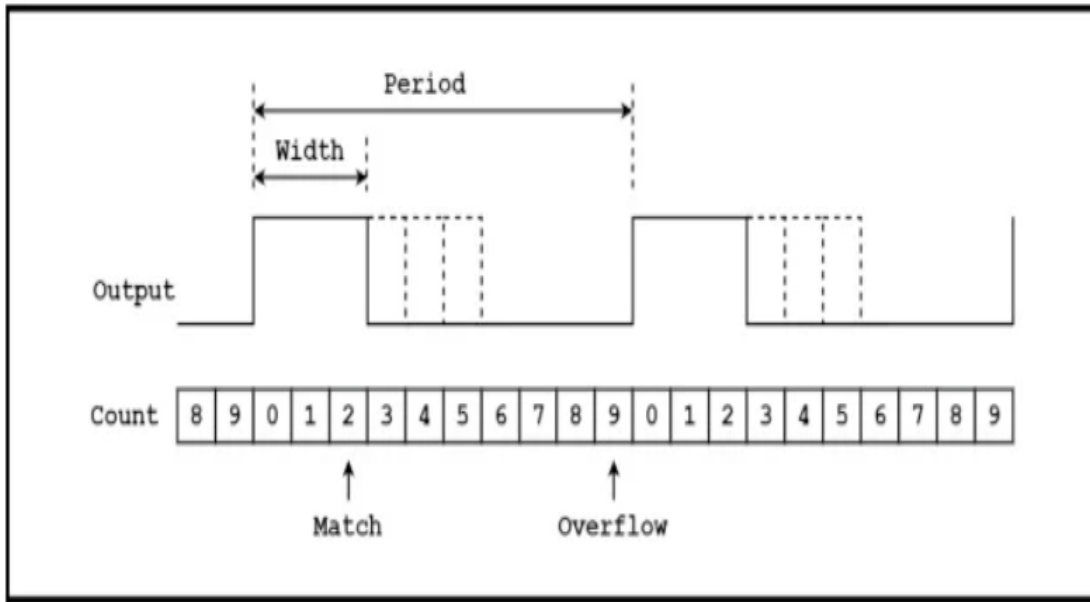


Figure 12: PWM Timing Diagram

PWM signal across RLC low pass Filter to invert DC signal

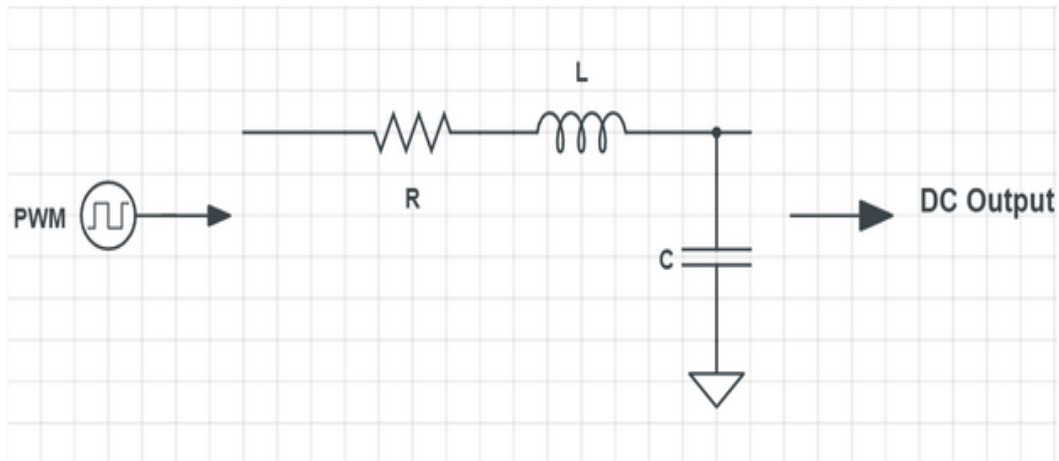


Figure 13: DC Voltage by using PWM

$$f_c = \frac{1}{2\pi\sqrt{LC}}$$

$$z = \sqrt{(X_L - X_C)^2 + R^2}$$

$$X_C = \frac{1}{2\pi fC}$$

$$X_L = 2\pi fL$$

5.1.5 Unity Gain Buffer

The output voltage follows or tracks the input voltage, the amplifier is a unity gain buffer. A voltage buffer amplifier's voltage gain may be unity, and it gives significant current gain.

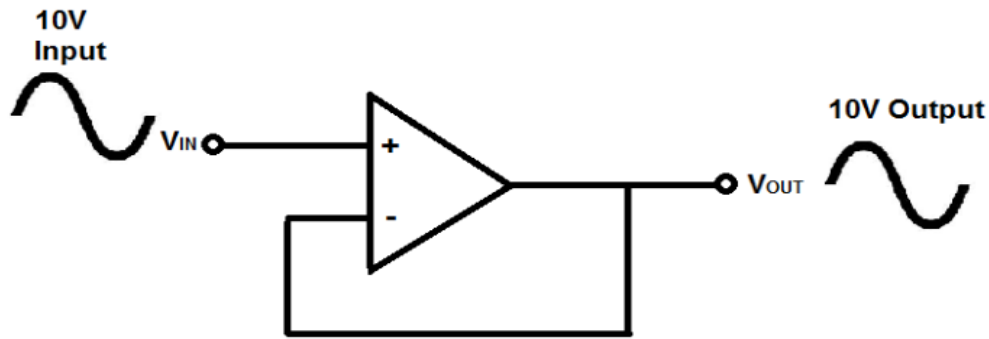


Figure 14: Unity Gain Buffer Configuration

5.1.6 RC Low Pass Filter

The low pass filter (LPF) is a filter that allows the passing of the signal with low frequency and blocks the signal with high frequencies [2]. Low pass filters are commonly used in electronic circuits to remove high frequency noise from a signal, to limit the bandwidth of a signal, or to isolate low frequency components of a mixed signal. Removing high frequency noise from a signal before amplification in order to improve sound quality.

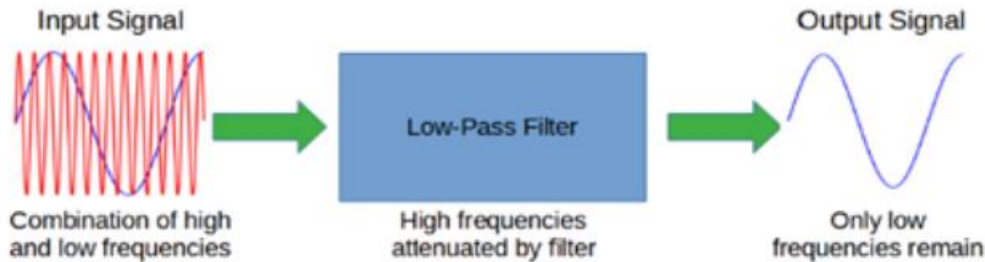


Figure 15: Low Pass Filter

$$f_c = \frac{1}{2\pi RC} \quad C = \frac{1}{2\pi R f_c} \quad R = \frac{1}{2\pi C f_c}$$

$$X_C = \frac{1}{2\pi f C} \quad z = \sqrt{X_C^2 + R^2} \quad V_{out} = \frac{A * X_C}{z}$$

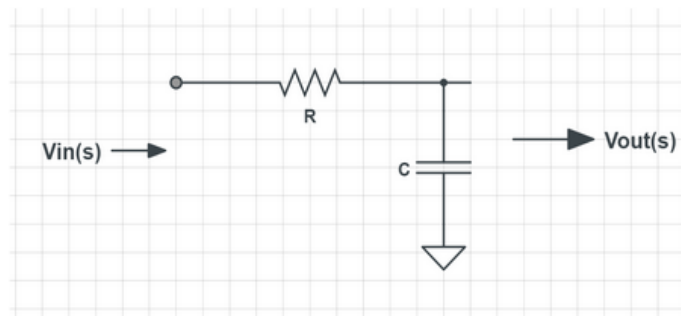


Figure 16: RC Low Pass Filter

5.2 MCU

- The microcontroller is like a computer integrated on a chip and used to control electronic devices. It is a self-contained embedded system with peripherals, processors, and memory.
- CPU is considered the brain of the microcontroller. It has the function of fetching and decoding instructions.
- Input/output port is used to control and communicate with devices such as monitors, LEDs, printers, etc.
- In microcontrollers memory is used to store data and programs.
- Serial port is the part that connects the microcontroller and other external peripherals.
- Timer and counter are a part of the clock speed of MCU, and it can be adjustable in configuration. It can be used as a delay or creating an event.

5.2.1 ADC Converters

In a digital oscilloscope, the analog signal is at the input, after attenuation/amplification and preliminary filtering such as for ac coupling, goes to an analog-to-digital converter (ADC). Each channel has a separate ADC, each with an external clock. Analog-to-digital conversion takes place by means of sampling. The samples are taken at specific rates and the amplitude of each sample is measured and stored. Intuitively, we know that the accuracy of the digital signal at the output depends upon the number of samples that are taken.

5.2.2 DMA

Direct Memory Access (DMA) is a capability provided by some computer bus architectures that allows data to be sent directly from an attached device (such as a disk drive) to the memory on the computer's motherboard. The microprocessor is freed from involvement with the data transfer, thus speeding up overall computer operation.

5.2.3 STM32CubeIDE

STM32CubeIDE is an all-in-one multi-OS development tool, which is part of the STM32Cube software ecosystem. STM32CubeIDE is an advanced C/C++ development platform with peripheral configuration, code generation, code compilation, and debug features for STM32 microcontrollers and microprocessors. It is based on the Eclipse/CDT framework and GCC toolchain for the development, and GDB for the debugging.

5.2.4 HAL Driver

The Hardware Abstraction Layer (HAL) driver layer provides a simple, generic multi-instance set of APIs (application programming interfaces) to interact with the upper layer (application, libraries and stacks). The HAL driver APIs are split into two categories: generic APIs, which provide common and generic functions for all the STM32 series and extension APIs, which include specific and customized functions for a given line or part number. The HAL drivers include a complete set of ready-to-use APIs that simplify the user application implementation. For example, the communication peripherals contain

APIs to initialize and configure the peripheral, manage data transfers in polling mode, handle interrupts or DMA, and manage communication errors. The HAL drivers are feature-oriented instead of IP-oriented. For example, the timer APIs are split into several categories following the IP functions, such as basic timer, capture and pulse width modulation (PWM). The HAL driver layer implements run-time failure detection by checking the input values of all functions. Such dynamic checking enhances the firmware robustness. Run-time detection is also suitable for user application development and debugging.

5.2.5 LL Driver

Low Layer (LL) drivers offer a fast light-weight expert-oriented layer which is closer to hardware than HAL. It means the LL driver offers low-level APIs at register level, such as a set of functions to initialize peripheral main features according to the parameters specified in data structures, a set of functions to reinitialize peripheral without recompiling. To use LL drivers, they require deep knowledge of the MCU and peripherals specifications.

5.3 GUI

For the GUI section, the GUI design is based on the PyQtGraph. When we connect the microcontroller board and front end to a PC via USB, then run a python script that we develop to launch the GUI. In this script, we will collect the live data using PySerial/PyUSB, process the data using Numpy, and we will plot the waveform using PyQtGraph. PyQtGraph is a pure-python graphics and GUI library built on PyQt/PySide and NumPy. It is intended for use in mathematics/scientific/engineering applications. Despite being written entirely in python, the library is very fast due to its heavy leverage of NumPy for number crunching and Qt's Graphics View framework for fast display. PyQtGraph is distributed under the MIT open-source license.

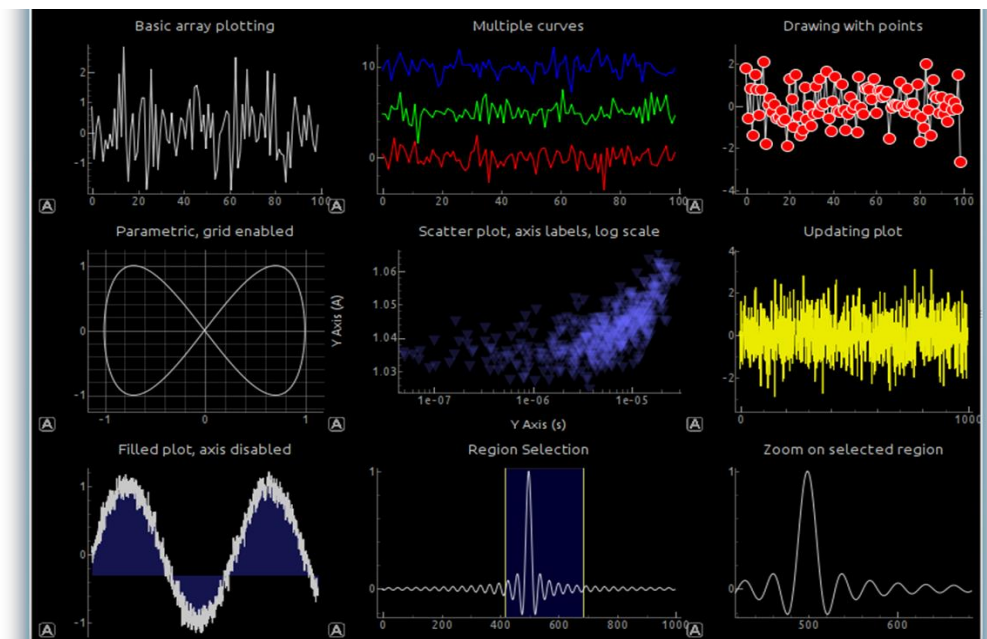


Figure 17: Main Features of PyQtGraph

PyQtGraph is known to run on Linux, Windows, and OSX. It should, however, run on any platform which supports the following packages:

- Python 3+
- PyQt 5, PyQt6, PySide2, or PySide6
- NumPy
- SciPy is optional for some numerical procedures
- python-opengl bindings are required for 3D graphics

PyQtGraph can be installed using these methods:

- From PyPI
 - Last released version: `pip install pyqtgraph`
- From conda
 - Last released version: `conda install -c conda-forge pyqtgraph`
- To install system-wide from source distribution: `python setup.py install`
- Many linux package repositories have released versions.
- To use with a specific project, simply copy the PyQtGraph subdirectory anywhere that is importable from your project.

We can learn PyQtGraph by browsing through the examples. We can use this command `python -m pyqtgraph.examples` to launch the example application.

6. Detailed Design

The information below presents a comprehensive design and calculation for a single channel oscilloscope. The design encompasses three main components: a front-end hardware component, a Microcontroller Unit (MCU) equipped with features like a timer and speed DMA, and a user-friendly GUI component. Each of these components is described in detail to provide a complete understanding of the oscilloscope's functionality.

6.1 Analog Front-End Schematics

The analog front-end circuit constitutes a vital component of a digital storage oscilloscope. This paper presents the design of a high-performance analog front-end circuit featuring wide band, low noise, and high input impedance characteristics. The circuit design encompasses several building blocks such as high impedance buffer, DC offset adjustment, and gain amplifier. A high-impedance passive probe was utilized to conduct an extensive performance evaluation of the circuit in both time and frequency domains.

6.1.1 Power Supply -5V Schematic

A -5 V power supply is a source of power that delivers a steady voltage output of -5 volts. It is typically used in electronic circuits that necessitate a dual power supply, meaning they require both a positive and negative voltage source to function such as a rail to rail amplifier in this design. To obtain the -5 V voltage output, the typical charge pump switching regulator is used in this design.

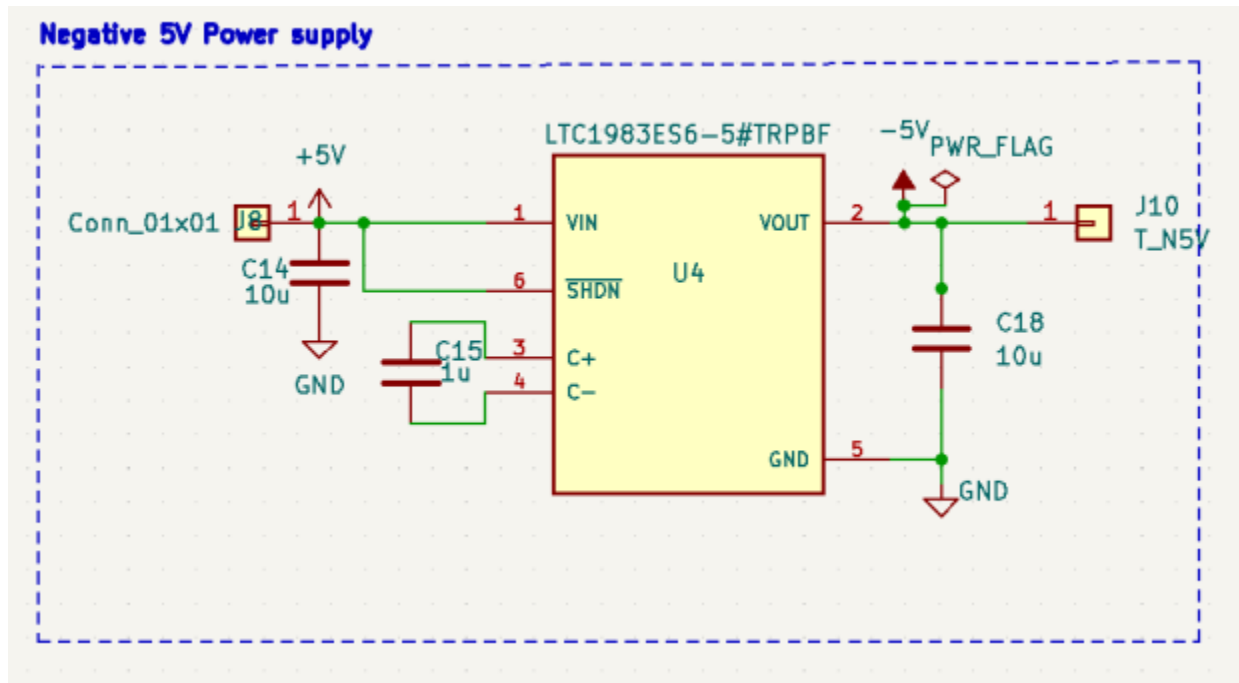


Figure 18: Negative 5V Power Supply Schematic

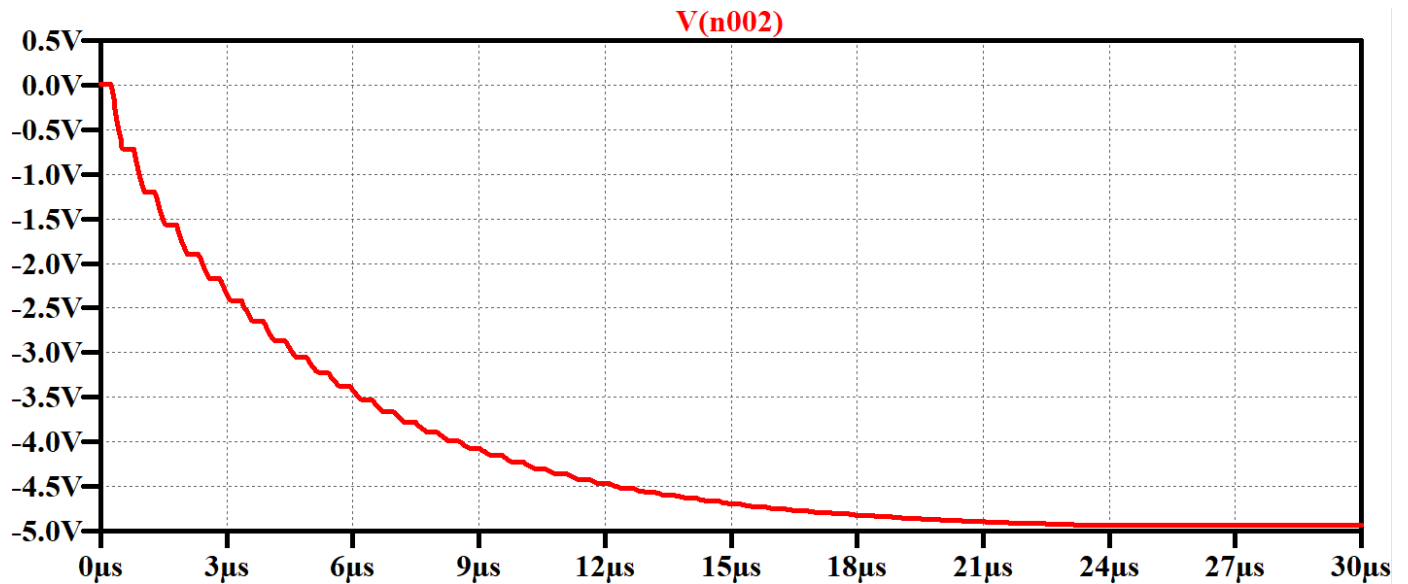


Figure 19: Negative 5V Power Supply Simulation

6.1.2 USB Soft Start Schematic

USB soft start serves the purpose of preventing current surges that occur when a USB device is first connected to a power source. These surges can potentially cause harm to either the USB device or the USB port. USB soft start achieves this by gradually ramping up the voltage across the USB port that the typical pmos transistor is using in the circuit for thus controlling the flow of current. This controlled flow of current ensures the safety of both the USB device and the USB port from any possible harm.

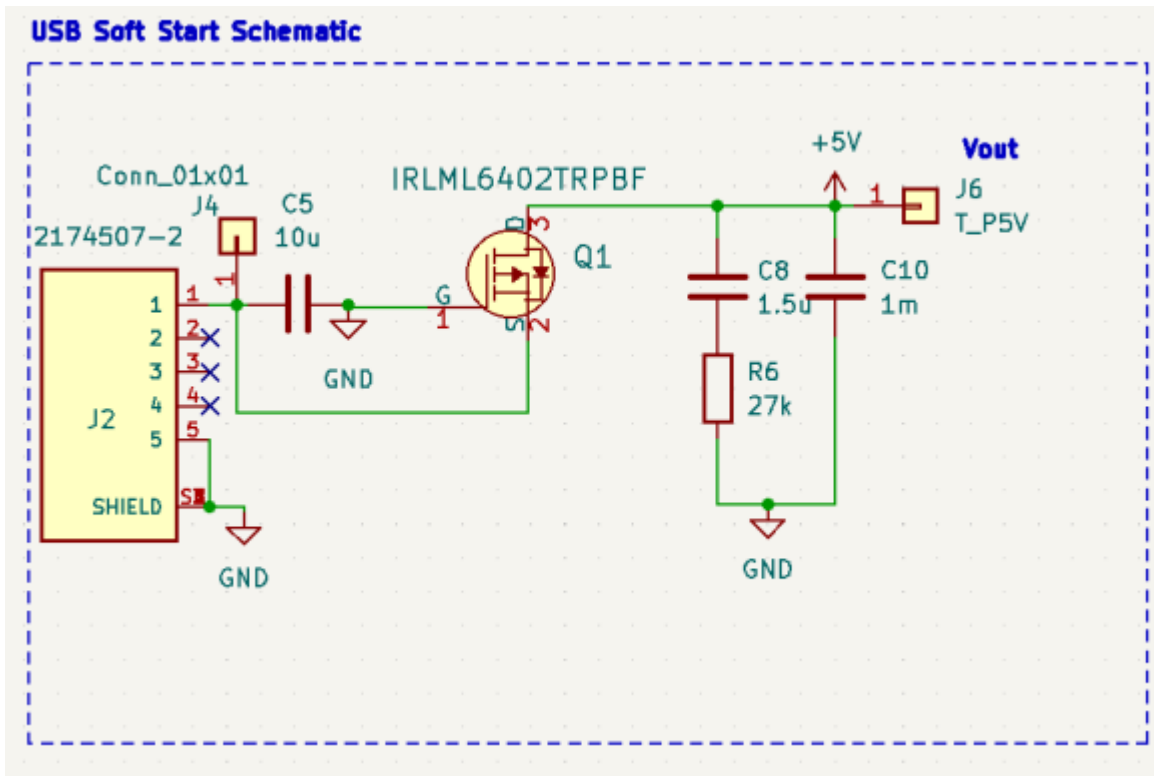


Figure 20: USB soft start schematic

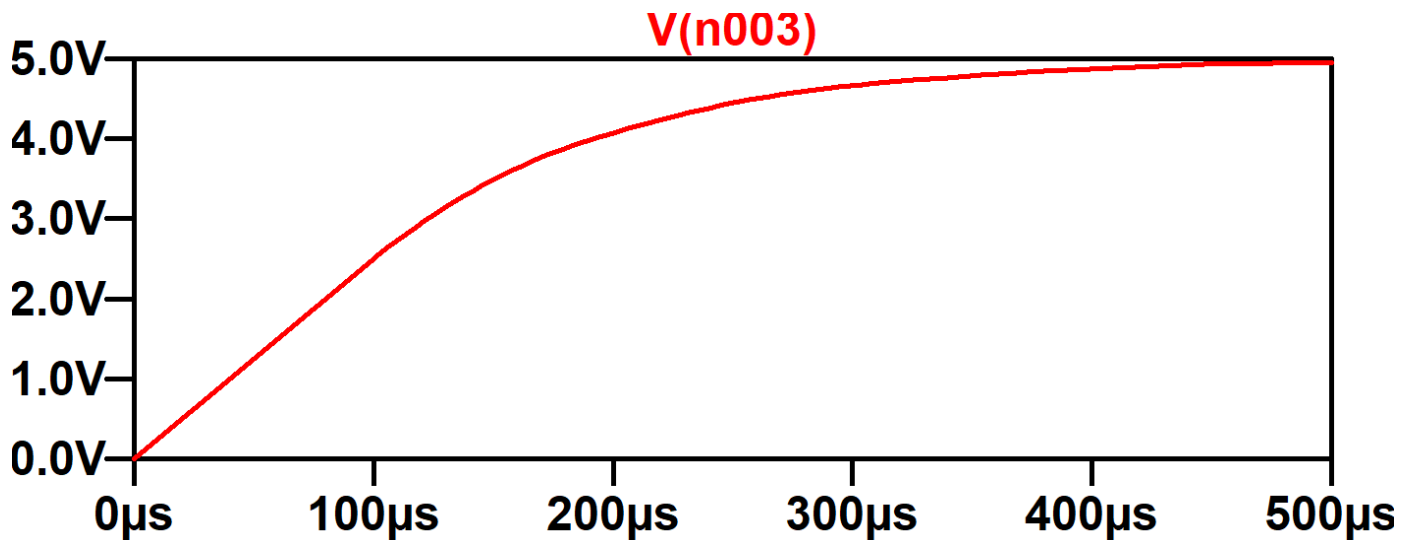


Figure 21: USB Soft Start Simulation

6.1.3 Attenuator Schematic

Our design utilizes mechanical relays due to several advantages they offer over digital switches, such as a high degree of isolation, affordability, and immunity to signal frequency. Specifically, mechanical relays are not affected by the signal frequency, ensuring consistent performance across a range of frequencies. This makes them an affordable option for applications where cost is a concern,

without compromising on performance. For the resistor and capacitor, the standard value with tolerance accepted (1 - 5%) for cost saving purposes.

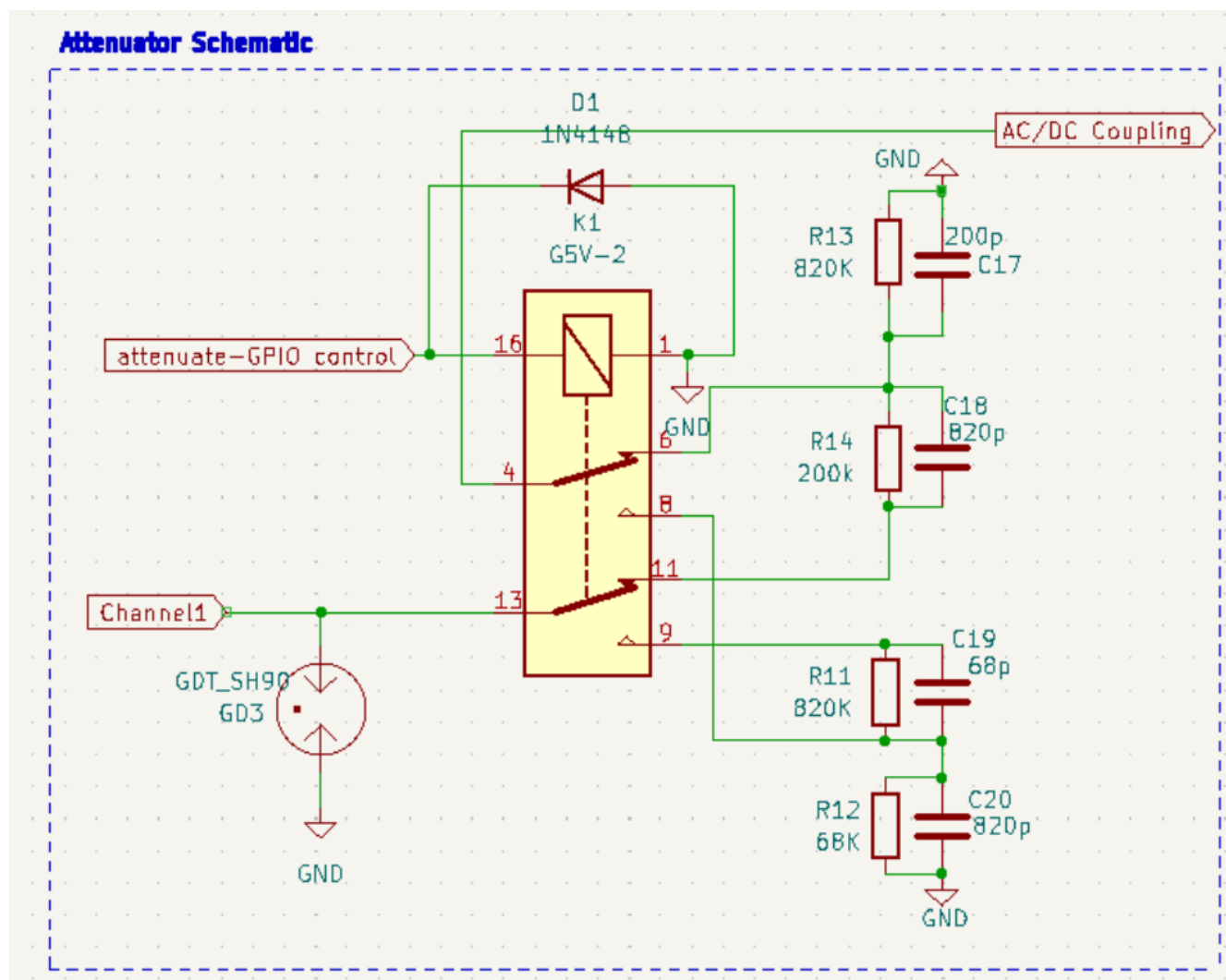


Figure 22: Attenuator Schematic

Calculation for (1.25:1) Scale

$$V_2 = V_s \frac{R_2}{R_1 + R_2} = V_s \frac{820k}{820k + 200k} = V_s * 0.8$$

Select: $R_2 = 820 k\Omega$

$$R_1 = 200 k\Omega$$

$$C_1 = 820pF$$

$$C_2 = 200pF$$

Calculation for (12.5:1) Scale

$$V_2 = V_s \frac{R_2}{R_1 + R_2} = V_s \frac{68k}{820k + 68k} = V_s * 0.08$$

Select: $R_2 = 68\text{ k}\Omega$

$R_1 = 820\text{ k}\Omega$

$C_1 = 68\text{ pF}$

$C_2 = 820\text{ pF}$

6.1.4 AC/ DC Coupling Schematic

The Photo MOS relay, also known as solid-state relays, offer several benefits over traditional mechanical relays is used in our design to optimize the component space, stable operation, low power consumption, and low noise.

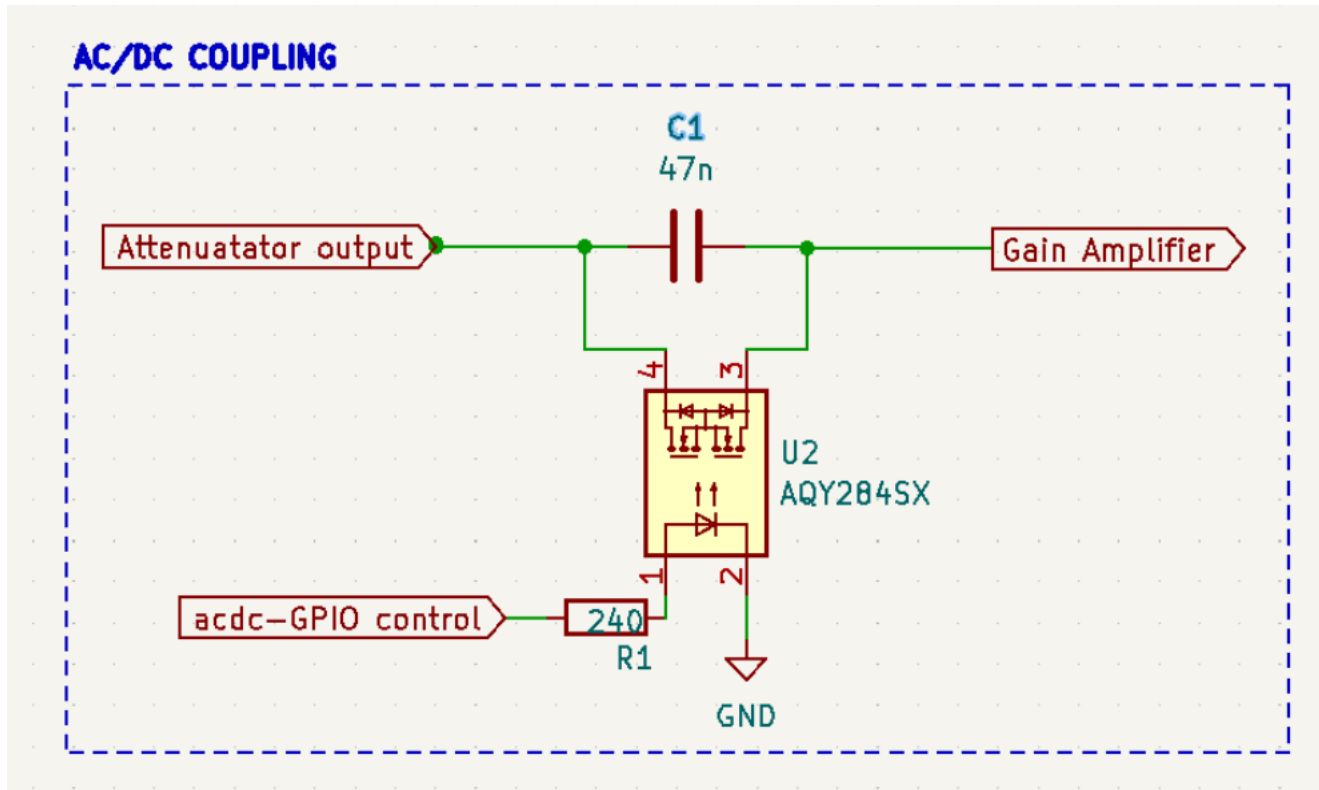


Figure 23: AC/DC Coupling Schematic

$$C = \frac{1}{2\pi f} = \frac{1}{2\pi * 5 * 10^5} = 318\text{ nF}$$

Select $C = 47\text{ nF}$ in order to have the signal over 500kHz pass through.

6.1.5 Gain Amplifier Schematic

In order to meet the input requirement for the ADC of Microcontroller which is working from 0 to 3.3 Voltage. The input signal will be divided into certain range as the table #1 and the different amplifier scale is designed such 1, 2.5, 5, 10 times in order to meet the output within 0-3.3 Voltage range. The maximum amplification is 10 and bandwidth requirement is 2 MHz, the op-amp LMP 7731 is selected for our design with the following features such as low noise, precision, with Gain Bandwidth Product (GBW) 22 MHz, Rail to Rail supply voltage 10V.

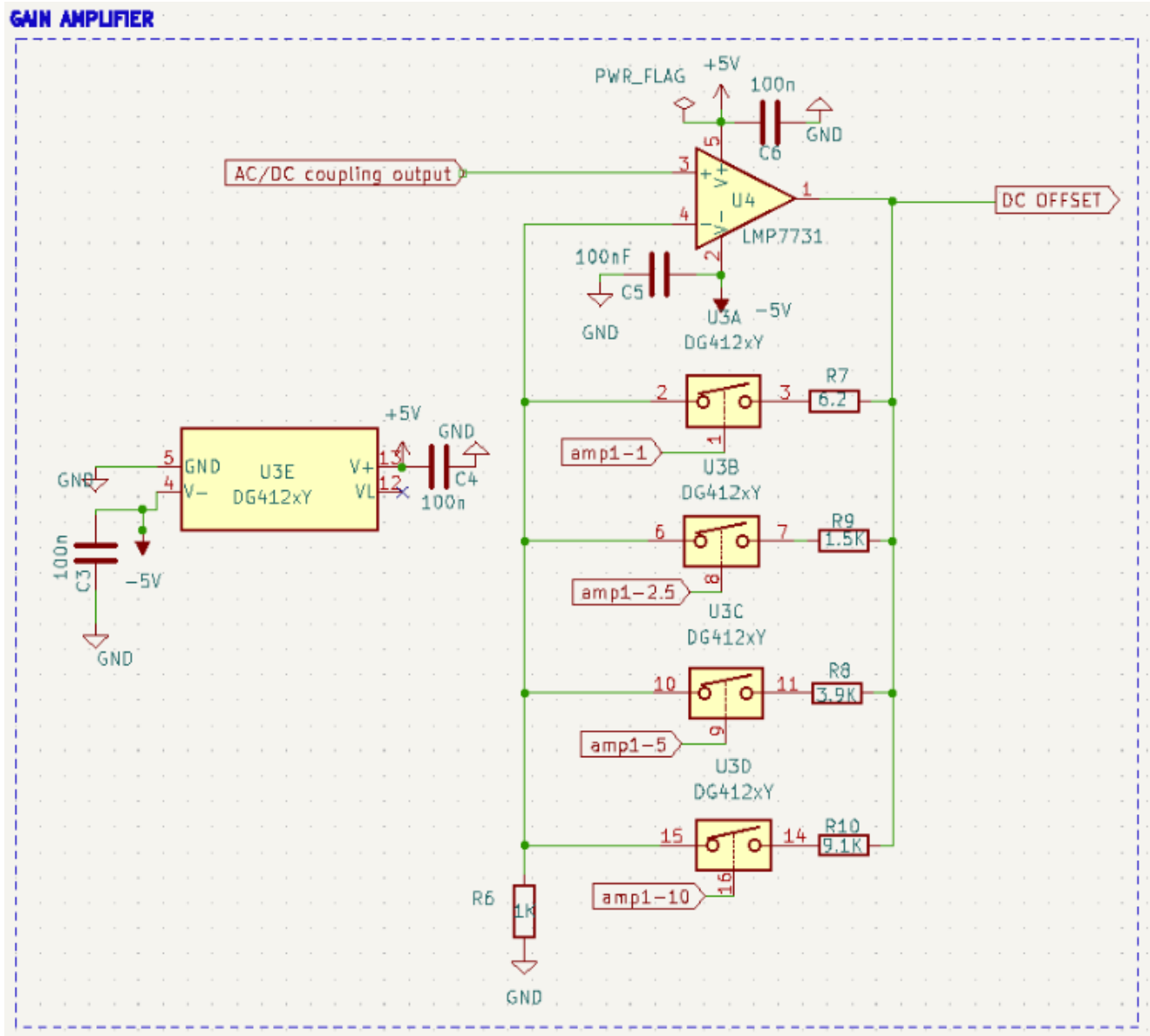


Figure 24: Gain Amplifier Schematic

Setting	Range	Peak to Peak	Attenuation		Amplification		Final
[V/Div]	[+/-V]	[V]		[V]		[V]	Amplification Value
0.01	0.04	0.08	1.25	0.06	10	0.64	8
0.02	0.08	0.16	1.25	0.13	10	1.28	8
0.05	0.2	0.4	1.25	0.32	10	3.20	8
0.1	0.4	0.8	1.25	0.64	5	3.20	4
0.2	0.8	1.6	1.25	1.28	2.5	3.20	2
0.5	2	4	1.25	3.20	1	3.20	0.8
1	4	8	13	0.62	5	3.08	0.38
2	8	16	13	1.23	2.5	3.08	1.19
5	20	40	13	3.08	1	3.08	0.077

Table 2: Amplification vs Voltage Range

Calculation:

- 1:1 Path

$$A_v = 1 + \frac{R_f}{R_2} = 1 + \frac{6.2}{1000} \sim 1$$

Select: $R_f = 6.2 \Omega$

$$R_2 = 1 k\Omega$$

- 1:2.5 Path

$$A_v = 1 + \frac{R_f}{R_2} = 1 + \frac{1500}{1000} = 2.5$$

Select: $R_f = 1.5 k\Omega$

$$R_2 = 1 k\Omega$$

- 1:5 Path

$$A_v = 1 + \frac{R_f}{R_2} = 1 + \frac{3900}{1000} \sim 5$$

Select: $R_f = 3.9 k\Omega$

$$R_2 = 1 k\Omega$$

- 1:10 Path

$$A_v = 1 + \frac{R_f}{R_2} = 1 + \frac{9100}{1000} \sim 10$$

Select: $R_f = 9.1 k\Omega$

$$R_2 = 1 k\Omega$$

6.1.6 DC Offset using PWM, Buffer, and RC Low Pass Filter Schematic

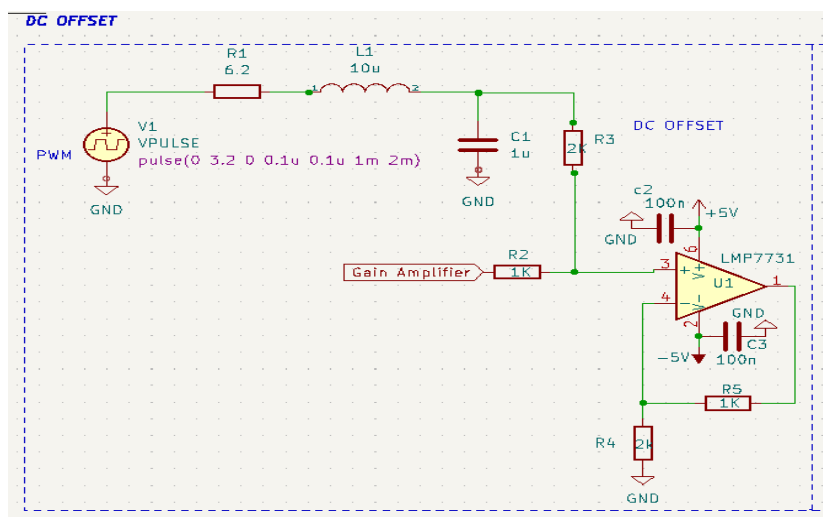


Figure 25: DC Offset using PWM, Buffer, and RC Low Pass Filter Schematic

PWM calculation:

In the case critically damped, $\zeta=1$ the following relation between R, L, C must be true:

$$R = 2 \sqrt{\frac{L}{C}}$$

Choose, $L = 10\mu\text{H}$ and $C = 1\mu\text{F}$, so $R = 6.2\Omega$ and $f_c = 50\text{ kHz}$.

The variable DC signal output depends on the duty cycle of the PWM signal.

6.1.7 Buffer and RC Low Pass Filter, and Voltage Protection Schematic

The typical unity amplifier is used here to strengthen the signal OPA863 is used in our design. In addition, Low Pass Filters are commonly used in electronic circuits to remove high frequency noise from a signal, to limit the bandwidth of a signal, or to isolate low frequency components of a mixed signal. They are also used in audio applications to remove high frequency noise from a signal before amplification or playback, resulting in improved sound quality.

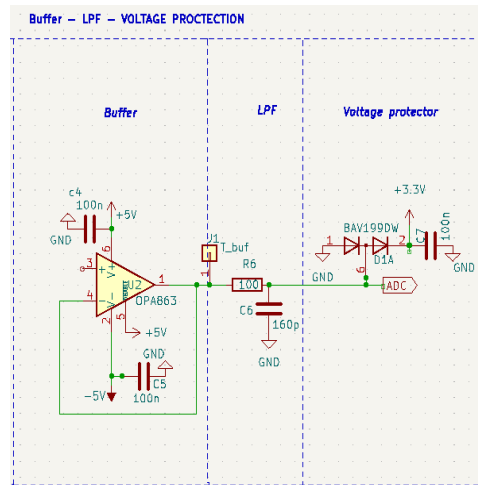


Figure 26: Buffer and RC Low Pass Filter, and Voltage Protection Schematic

Low pass filter calculation:

frequency	Xc	Z	Vout
10	99471839.43	99471839.43	5
50	19894367.89	19894367.89	5
100	9947183.943	9947183.944	5
1000	994718.3943	994718.3994	4.999999975
4000	248679.5986	248679.6187	4.999999596
10000	99471.83943	99471.8897	4.999997473
50000	19894.36789	19894.61921	4.999936836
100000	9947.183943	9947.686585	4.999747357
500000	1989.436789	1991.948477	4.993695398

Table 3: Frequency vs V_{out} of low pass filter simulation

$$f_c = \frac{1}{2\pi RC}$$

Chose $f_c = 10\text{MHz}$, $R = 100\Omega$, so $C = 160\text{pF}$

6.1.8 The Whole Design Schematic

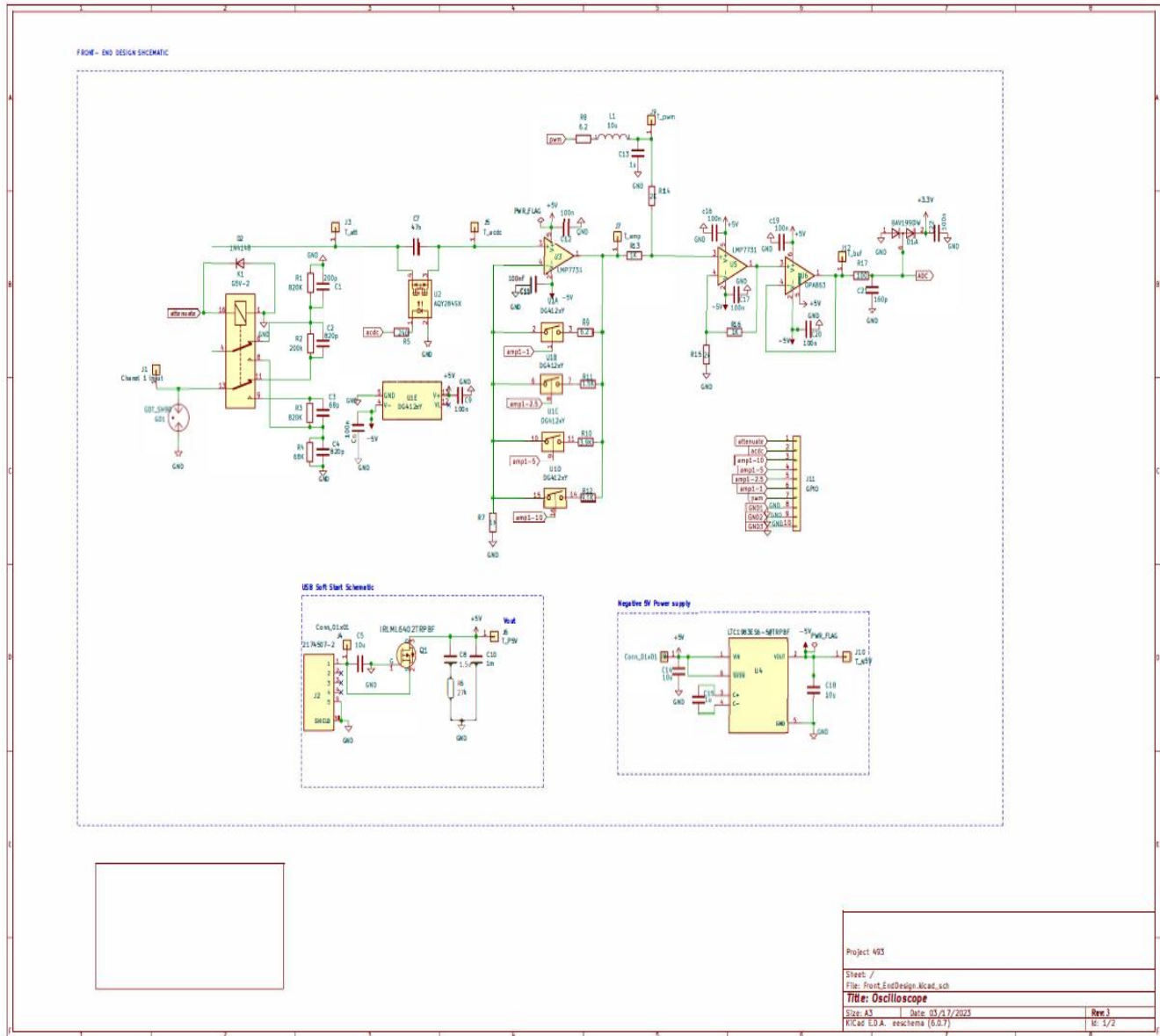


Figure 27: The Whole Design Schematic

6.2 PCB Analog Front-End

The Front-End Printed Circuit Board (PCB) has been designed with detail to ensure optimal signal transmission without any degradation or interference. The layout has been optimized to ensure that the input and output signals take the shortest and straightest path possible. To achieve this, the PCB has been designed with four layers, with the front and back layers dedicated to signals, an inner layer for power, and another layer for ground.

To create the PCB, KICAD 6 software was used to optimize the layout to a compact size of 2.71" x 1.44". OSH PARK manufacturer was selected to fabricate the PCB using FR408 substrate, purple mask over bare copper, and an ENIG (immersion gold) finish. The detailed design of the PCB, including all traces and vias, can be found in the table below. To accommodate the SMD components, 0603 resistor and capacitor footprints were used, which allows a trace to pass through the component. The assembly of the components was carried out in the GMU lab, with the use of a soldering oven for SMD components, and hand-soldering for through-hole components.

Description	Size
Trace clearance	5mil (0.127mm)
trace width for signal	12mil (0.305mm)
Trace with for power and GND	24 mil(0.6096mm)
Drill size	10mil (0.254mm)
Annular ring	4mil (0.1016mm)
Minimum Hole size	10 mil (0.254mm)
Maximum Drilled Hole Size	260 mil (6.604mm)
Via Plating Thickness	12mil (0.305mm)
Fabricated Hole Size Tolerance	+/-2.5mil max (0.0635mm) +/-1 mil typical (0.0254mm)
Fabricated Hole Position Tolerance	+/-2.5mil max (0.0635mm) +/-1 mil typical (0.0254mm)

Table 4: PCB Design Detail

J11 (connects to MCU)	
1	Attenuator
2	AC/DC
3	Amp 1-10
4	Amp 1-5
5	Amp 1-1.25
6	Amp 1-1
7	PWM
8 - 9 - 10	GND

Table 5: J11 Pinout Detail connects to MCU

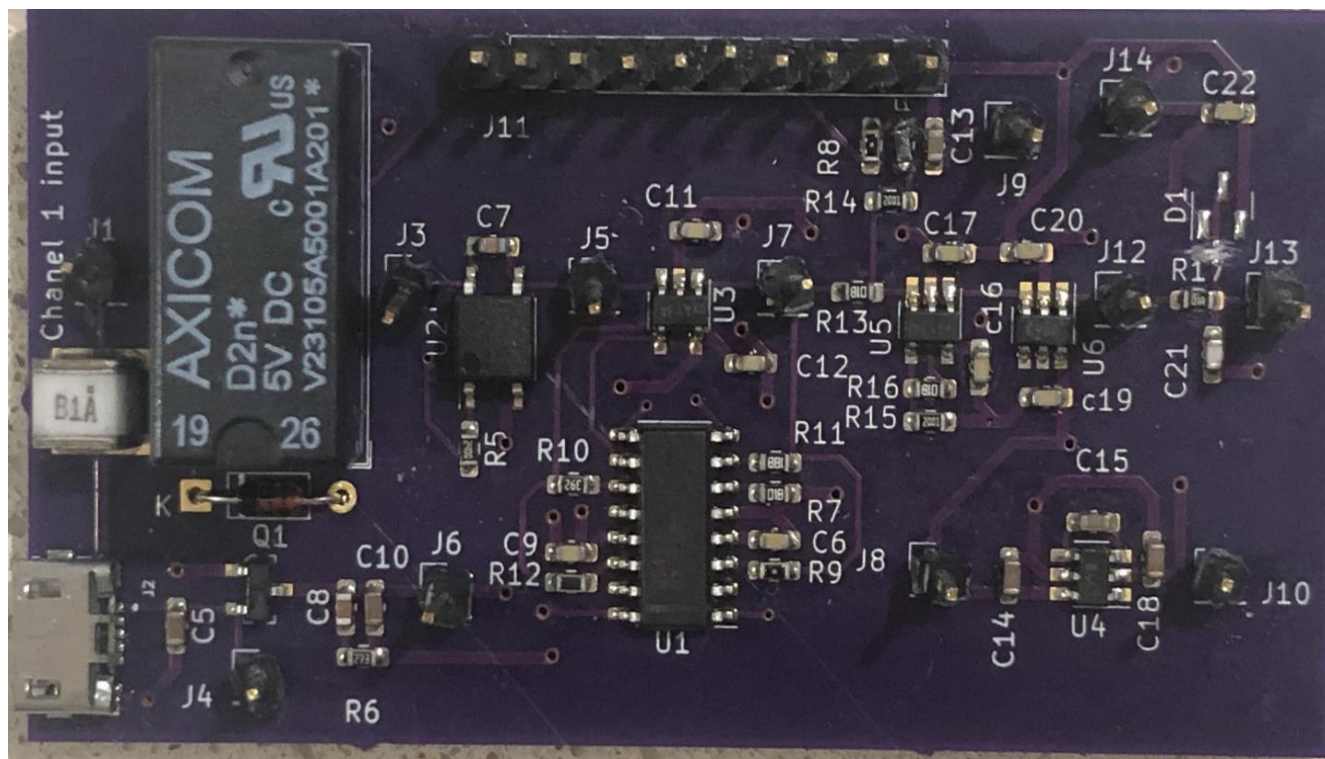


Figure 28: Front of PCB for Single Channel

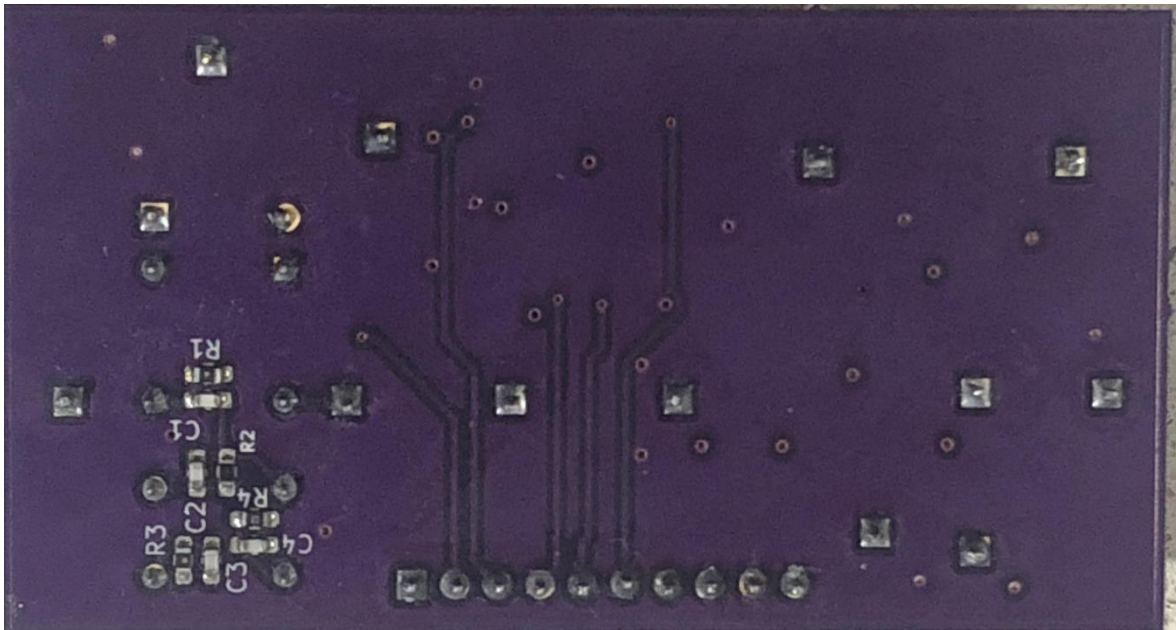


Figure 29: Back of PCB for Single Channel

6.3 MCU Design

For the detailed design of MCU, there are three parts. First, the controller layer is the microcontroller processor configuration. The function of the MCU is required for the GUI. This function will include direct memory access (DMA), ADC converter and speed, Timer, GPIO and the USB transmit/receive. STM32CubeMX and Keil will be used to support these functions. STM32CubeMX is used for setup MCU configuration, and Keil is used for coding part of the MCU. For the MCU hardware, this will include the Hardware abstraction layer (HAL) driver and the low layer (LL) driver. HAL driver is used to perform functions such as GPIO, Timer, ADC, DMA, and USB. LL driver is used for adjusting or changing in register level while the MCU is running without reinitialization by STM32CubeMX.

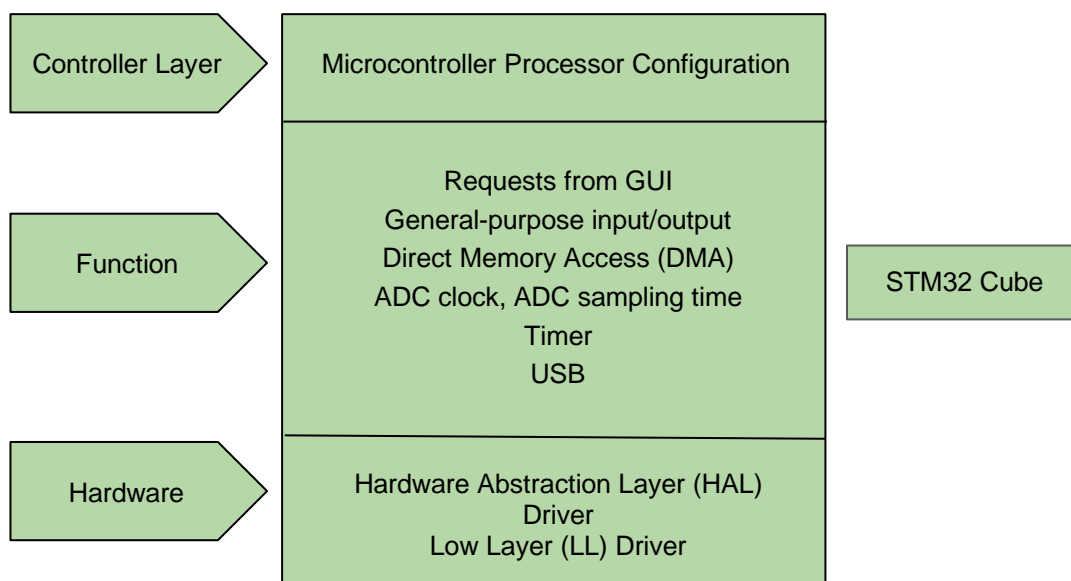


Figure 30: MCU Detailed Design

Connection					
Pin Name	Purpose (Hardware)	Purpose (Software)	Pin Name	Purpose (Hardware)	Purpose (Software)
PA0		ADC1_In	PA12		USB DP
PA8 (1/0)	Amplifier 1:10 (on/off)	GPIO_Out	PB4 (1/0)	Amplifier 1:1 (on/off)	GPIO_Out
PA9 (1/0)	AC/DC and AC	GPIO_Out	PB10 (1/0)	Amplifier 1:2.5 (on/off)	GPIO_Out
PA10 (1/0)	Attenuator (12.5/1.25)	GPIO_Out	PC7 (1/0)	Amplifier 1:5 (on/off)	GPIO_Out
PA11		USB DM			

Table 6: IO Pins Connect to Front_End Design

These are pin names and pin purposes we use for 1 channel of Front-End design. We use 6 pins (PA8-10, PB4, PB10, and PC7) with GPIO_Out purpose to change settings in the design. We use 1 pin (PA0) as the input which is the output from Front-End design. We use 2 pins (PA11, PA12) for USB data minus and USB data plus, and they are connected between the microcontroller and PC.

Pin Configuration:

The STM32F303RE microcontroller is a versatile and powerful device that is commonly used in a wide range of embedded systems applications. One of the key features of this microcontroller is its ability to operate at a maximum clock speed of 72 MHz, which allows it to process instructions at a high rate of speed.

USB: We need to enable USB from STM32CubeIDE. In USB_DEVICE, we need to choose Communication Device Class (Virtual Port Com). Then, we need to connect an external 8.000 MHz crystal to the board. After finishing configuration on STM32CubeIDE, we run the C file in STM32, then we can see the PC recognize the USB port in the PC device manager. To call USB functions, we need to include usbd_cdc_if.h in the C file.

Timer for PWM: we use Timer1 and channel1, we set it to PWM Generation CH1. From the clock source, we choose an internal clock. In the parameter setting, we set the prescaler to 0, counter mode is up mode, counter period is 3300-1 (because we want to generate PWM from 0V to 3.3V). Also, in DMA settings, we need to add DMA corresponding to TIM1_CH, change direction Memory to Peripheral.

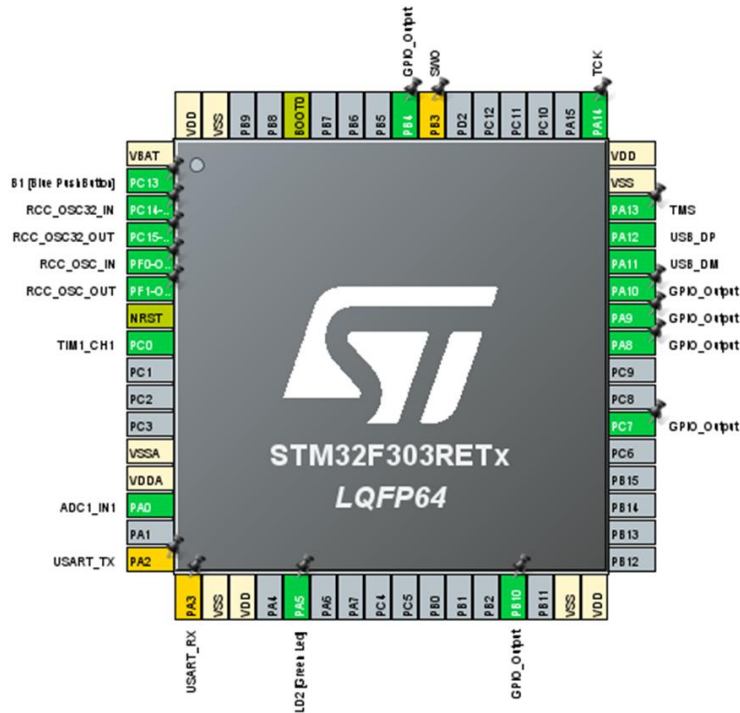


Figure 31: Pin Configuration for Front-End

Timer for trigger: we need to activate Timer16, and set prescaler 72-1 (72 Mhz to 1Mhz), counter up mode. Also, we enable Timer16 interrupt to perform tasks in interrupt service routine.

ADC: we choose ADC1, input 1 single-ended. ADC 12-bit resolution, right alignment, enable continuous conversion mode, enable DMA continuous requests. The rest is the default setting.

DMA: in the DMA option, we need to add ADC1, direction is peripheral to memory, and circular mode.

GPIO: To achieve the desired functionality of the Front-End design, we need to configure the GPIO pins as output pins. This configuration enables the microcontroller to control the signals that are sent out to the Front-End components. It is essential to carefully select the specific GPIO pins to be used for this purpose, based on their availability and suitability for the Front-End design. Once the appropriate pins have been identified, we can proceed to configure them as output pins by setting the GPIO_Out flag. This configuration enables the microcontroller to drive the selected pins with the appropriate signals required by the Front-End design. By setting up the GPIO_Out flag correctly, we can ensure that the Front-End design functions correctly and reliably.

RCC: RCC, which stands for "Resistor-Capacitor-Crystal," is a crucial component in electronic circuits that provides clock signals to various digital components. To ensure proper functioning of the circuit, it is essential to set up both high-speed and low-speed clocks, which can be achieved through the use of crystal/ceramic resonators. These resonators are highly accurate and stable devices that generate precise oscillations at a particular frequency, which can be used as a clock signal for the digital components. By carefully selecting and setting up these resonators, the RCC can maintain accurate and reliable clock signals, ensuring that the electronic circuit functions optimally.

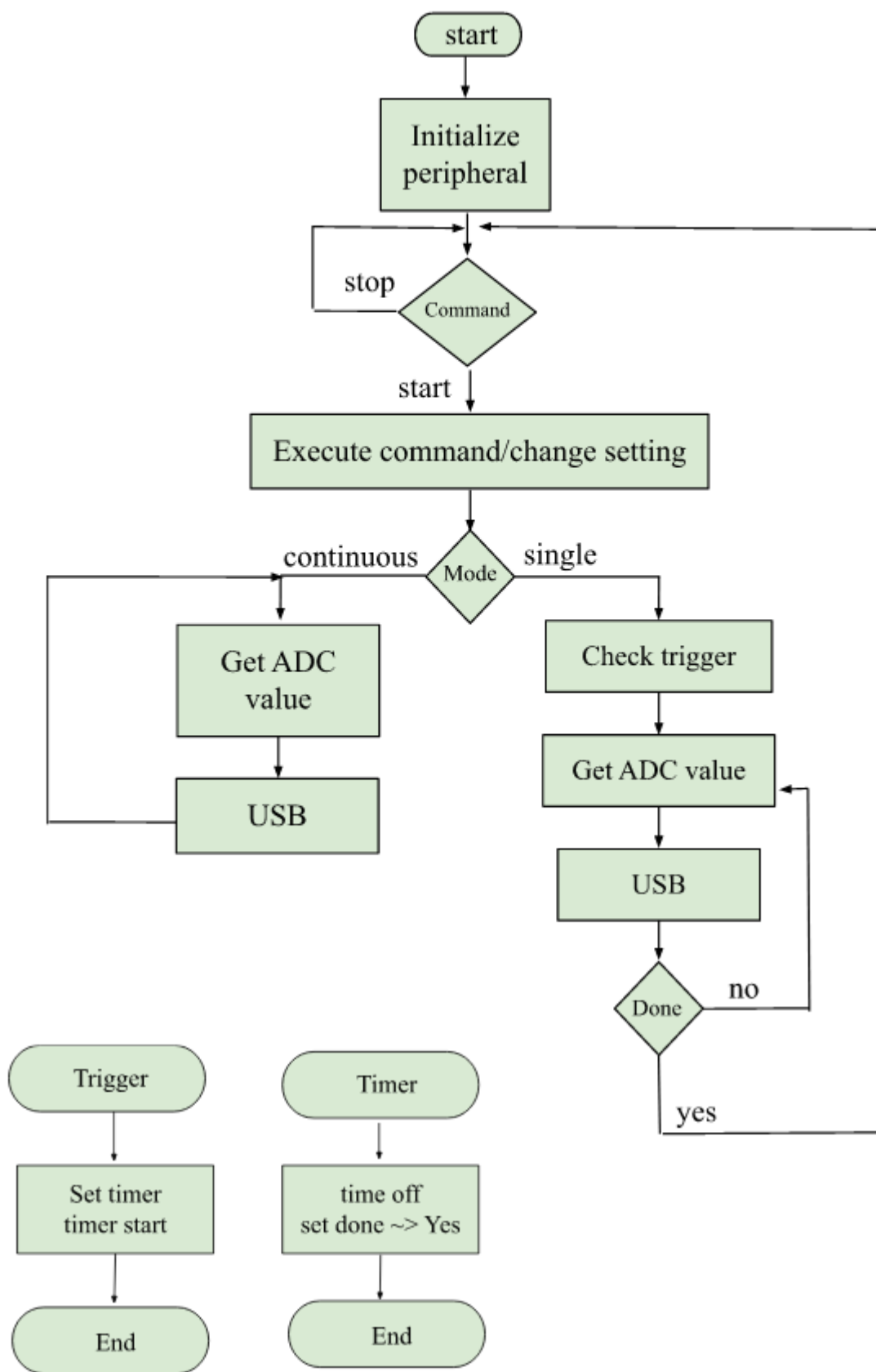


Figure 32: MCU Flow Chart

When starting the device, the device is initialized peripheral, then moved to the command block. In the command block, the device checks the command coming from the PC, either stop or start command. If the command is stop, the device stops everything and waits for the next command. If the command is start, the device executes the adjust setting, then moves to mode block. The command from the PC will tell the device to perform continuous mode or single mode. In the continuous mode, the device gets ADC data and sends it to the PC by USB. The device keeps performing that cycle until it receives a new command. When the device is in single mode, it checks the trigger and sets a timer to know when to stop sending data to the PC and wait for the next command.

The below table is calculated by the following formula and based on fixed samples (10000 samples). The sampling time and ADC clock need to be changed depending on input and screen division to display the signal correctly. Otherwise, if the ADC reads too fast, the signal is up and down because data is overwritten. If the ADC reads too slow, the data is repeated.

$$\text{Sample time} = \frac{\text{screen}}{10000} \times \text{ADC Clk} - 12.5$$

$$\text{Sample per second} = 1 \div \frac{\text{sample time} + 12.5}{\text{ADC Clk}}$$

$$T_{\text{conv}} = \frac{\text{sample time} + 12.5}{\text{ADC Clk}}$$

Time per division	Screen	Sample per second	Samples	ADC Clock	Sample time	Calculated sample time
1us	10us	5Msps	50	72Mhz	1.5	
2us	20us	5Msps	100	72Mhz	1.5	
5us	50us	5Msps	250	72Mhz	1.5	
10us	100us	5Msps	500	72Mhz	1.5	
20us	200us	5Msps	1000	72Mhz	1.5	
50us	500us	5Msps	2500	72Mhz	1.5	
100us	1ms	5Msps	5000	72Mhz	1.5	
200us	2ms	5Msps	10000	72Mhz	1.5	
500us	5ms	2Msps	10000	36Mhz	4.5	< 5.5
1ms	10ms	1Msps	10000	18Mhz	4.5	< 5.5
2ms	20ms	0.5Msps	10000	9Mhz	4.5	< 5.5
5ms	50ms	225Ksps	10000	7.2Mhz	19.5	< 23.5
10ms	100ms	97Ksps	10000	7.2Mhz	61.5	> 59.5
20ms	200ms	56.25Ksps	10000	1.125Mhz	7.5	< 10
50ms	500ms	20.09Ksps	10000	0.28125Mhz	1.5	= 1.5625
100ms	1s	8.79Ksps	10000	0.28125Mhz	19.5	> 15.625
200ms	2s	3.80Ksps	10000	0.28125Mhz	61.5	> 43.75
500ms	5s	1.83Ksps	10000	1.125Mhz	601.5	> 550
1s	10s	916sps	10000	0.5625Mhz	601.5	> 550
Sample time options: [1.5, 2.5, 4.5, 7.5, 19.5, 61.5, 181.5, 601.5]						
ADC clk options(72Mhz): [72, 36, 18, 12, 9, 7.2, 6, 4.5, 2.25, 1.125, 0.5625, 0.28125]						

Table 7: Calculated sampling time and ADC clock

6.4 GUI Design

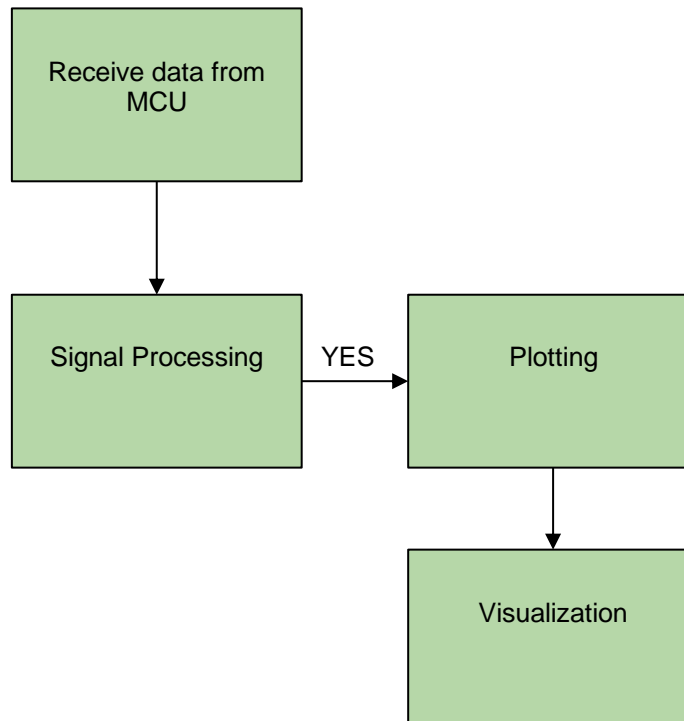


Figure 33: GUI Detailed Design

The goal for the GUI is to be able to process all the signals from the ADC to the original signal, create an array from that data, plot all that information fast enough to not notice any delay, and finally to be able to do all that with two-channels and a functional looking GUI. The GUI will be made fully with Python including some important math and GUI libraries, and these libraries are as follows:

- NumPy/SciPy (arrays and matrix math)
- Serial
- PyQt5/PySide (GUI toolkit)
- PyQtGraph (fast plotting of data)

These are the main libraries that are used to make the user interface of the system. The GUI allows the user to interact with MCU to display and manipulate oscilloscope readings. The GUI settings allow the user to adjust the voltage and time per division, apply a DC offset, and control the attenuator, amplifier, sampling time, and ADC clock settings. The voltage values displayed on the GUI are adjusted based on these settings, and the interaction between the GUI and the MCU involves scaling and inverting the voltage values to match the user-selected parameters. The GUI was tested using Windows 10 and 11, and for it to run properly, the port and baudrate number need to match the MCU's serial. The board also needs to be connected for it to work. to change the port and baudrate number, change `reader = SerialReader(port='COM7', baudrate=12000000)` in line 400 of the code. COM7 can be changed to COM# depending on the port number in the Device Manager of Windows. The main function `"def update():"`, and `"def plot_data():"` is the main loop that collects the data and plots it.

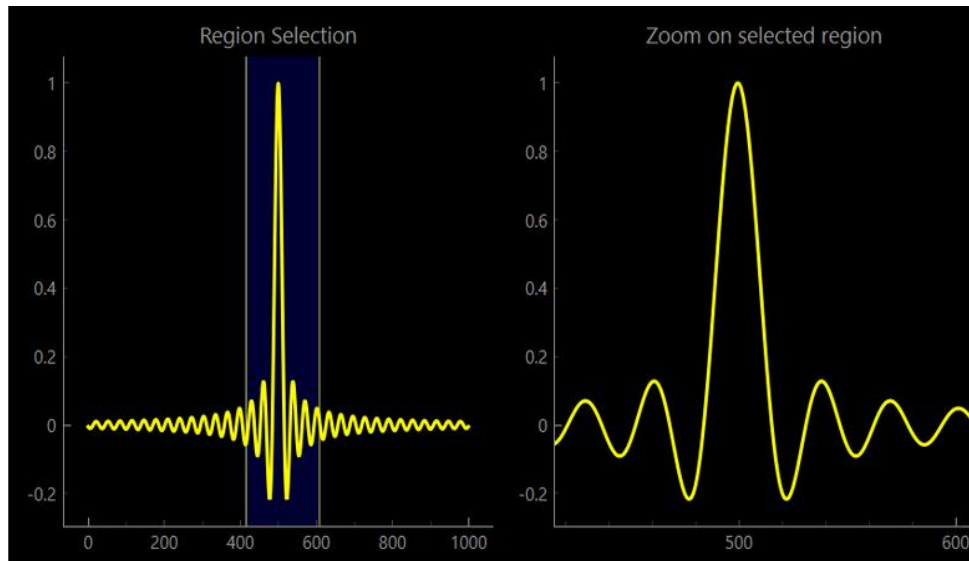


Figure 34: The zoom function in PyQtGraph

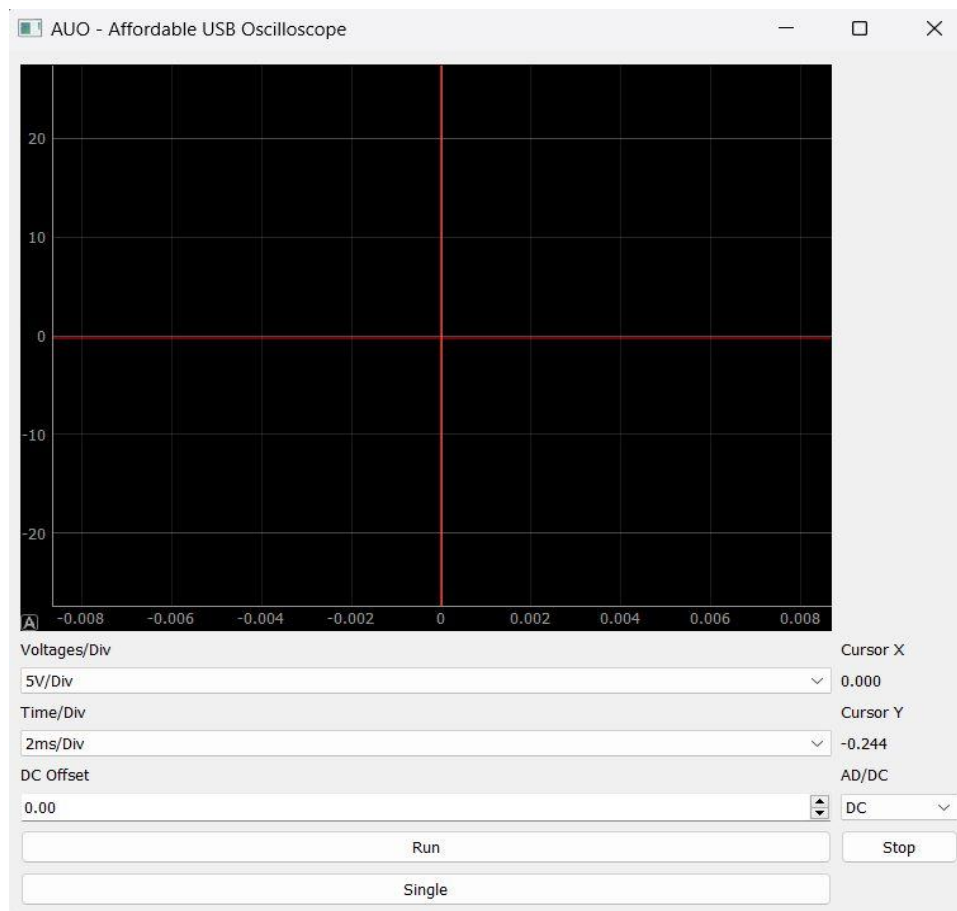


Figure 35: The GUI Screen

The GUI is made of three different sections: signal plot, controls, and cursor information.

1. Signal plot: The main part of the GUI is a plot displaying the oscilloscope readings in real-time or as a single capture, depending on the user's choice.

2. Controls: There are several control elements in the GUI, allowing users to adjust the display and functionality of the oscilloscope. These controls include:

- Voltages/Div: A combo box for selecting the voltage per division on the y-axis.
- Time/Div: A combo box for selecting the time per division on the x-axis.
- DC Offset: A spin box for adding an offset value to the voltage readings.
- Run and Stop buttons: Buttons for starting and stopping the continuous data reading and plotting.
- Single button: A button for capturing and plotting a single set of 200 points.
- Attenuator: A combo box for selecting the attenuator value.
- AD/DC: A combo box for selecting AC or DC coupling.
- Amplifier: A combo box for selecting the amplifier value.
- Sampling Time: A combo box for selecting the sampling time.
- ADC Clock (MHz): A combo box for selecting the ADC clock frequency.

3. Cursor: The plot includes a crosshair cursor that moves with the mouse and displays the x and y values at the cursor's position.

The received data from the MCU is scaled to voltage values between 0 and 3.3 volts. The received values are then adjusted based on the user-selected settings for the attenuator and amplifier that are determined by the selected attenuator and amplifier settings. The calibration process involves adjusting the displayed values based on the selected GUI settings. This is achieved by applying the DC offset value and inverting the original voltage according to the attenuator and amplifier settings.

7. Experimentation and Success Evaluation

7.1 Experimentation of MCU

7.1.1 Experiment 1: USB

In order to prepare for the PCB-USB testing later on, the first step is to figure out how to use the STM32 and turn it into a USB that can transfer the data. This experiment is dependent on the coding in IDE and a simple circuit that is connected to the STM32. To make sure that the USB can transfer data, a message is being added to the code section. After connecting the USB wire to the computer and running the code for STM32, Hercules SETUP utility is used to view whether or not the message can be transferred successfully.

```
while (1)
{
    /* USER CODE END WHILE */
    // transfer message from MCU to laptop
    CDC_Transmit_FS((uint8_t *)data, strlen(data));
    HAL_Delay(1000);
    // receive message from laptop to MCU
    // if message == 'o', LED on
    if (buffer[0]=='o'){
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5,SET);
    }
    // if message == 'z', LED off
    else if (buffer[0]=='z'){
        HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5,RESET);
    }
}
/* USER CODE BEGIN 3 */
```

Figure 36: The code is for testing with receiving, transferring, and turning on/off LED

```

9 static int8_t CDC_Receive_FS(uint8_t* Buf, uint32_t *Len)
10 {
11     /* USER CODE BEGIN 6 */
12     USBD_CDC_SetRxBuffer(&hUsbDeviceFS, &Buf[0]);
13     USBD_CDC_ReceivePacket(&hUsbDeviceFS);
14     uint8_t len = (uint8_t) *Len;
15     memcpy(buffer, Buf, len);
16     return (USBD_OK);
17     /* USER CODE END 6 */
18 }

```

Figure 37: USB Receive Function

For transferring, we do not need to do anything. To receive data, we need to modify the receive function of the board. We need to add a buffer, so data coming from the PC will be stored in it. Then MCU will change performance based on how we design MCU.

```

while (1)
{
    if (n==10240){
        n=0;
        HAL_ADC_Start_DMA(&hadc1, (uint32_t*)adc_buff, ADC_BUFF);
    }
    sprintf(msg, "%hu\r\n", adc_buff[n]);
    CDC_Transmit_FS((uint8_t *)msg, strlen(msg));
    HAL_Delay(1);
    n++;

    if (command_buffer[0]=='1'){ // command string is new?
        if (command_buffer[1]=='0'){ // SMP 1.5 clock cycles
            changeSampling0(&hadc1);
            command_buffer[0]='0'; // set this command string becomes old
        }
        else if (command_buffer[1]=='1'){ // SMP 2.5 clock cycles
            changeSampling1(&hadc1);
            command_buffer[0]='0';
        }
        else if (command_buffer[1]=='2'){ // SMP 4.5 clock cycles
            changeSampling2(&hadc1);
            command_buffer[0]='0';
        }
        else if (command_buffer[1]=='3'){ // SMP 7.5 clock cycles
            changeSampling3(&hadc1);
            command_buffer[0]='0';
        }
        else if (command_buffer[1]=='4'){ // SMP 19.5 clock cycles
            changeSampling4(&hadc1);
            command_buffer[0]='0';
        }
        else if (command_buffer[1]=='5'){ // SMP 61.5 clock cycles
            changeSampling5(&hadc1);
            command_buffer[0]='0';
        }
    }
}

```

Figure 38: Combining Transferring, Receiving by USB and ADC

The code is MCU transfers data to laptop, and MCU receives data from laptop to change MCU configuration. The code is MCU performs ADC. If MCU receives data coming from the PC, MCU will change ADC speed such as sampling time or ADC clock to slow down or speed up ADC.

7.1.2 Experiment 2: GPIO

MCU also needs to use STM32 to control the 3.2V DC voltage input for Front-End. Before plugging in to the Front-End breadboard circuit, testing was done through a simple LED circuit. By using the GPIO on the STM32, the code is modified to be high or low in order to turn on and off the LED on the breadboard. The testing was working well. After that, STM32 was connected directly to the breadboard circuit of Front-End for DC and attenuator testing. When modifying the code on STM32, the STM32 successfully provided 3.2V DC voltage to the circuit.

```

69 int main(void)
70 {
71     /* USER CODE BEGIN 1 */
72
73     /* USER CODE END 1 */
74
75     /* MCU Configuration-----*/
76
77     /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
78     HAL_Init();
79
80     /* USER CODE BEGIN Init */
81
82     /* USER CODE END Init */
83
84     /* Configure the system clock */
85     SystemClock_Config();
86
87     /* USER CODE BEGIN SysInit */
88
89     /* USER CODE END SysInit */
90
91     /* Initialize all configured peripherals */
92     MX_GPIO_Init();
93     MX_USART2_UART_Init();
94     /* USER CODE BEGIN 2 */
95
96     /* USER CODE END 2 */
97
98     /* Infinite loop */
99     /* USER CODE BEGIN WHILE */
100    while (1)
101    {
102        // HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13); //PC13 Button
103
104
105        HAL_GPIO_WritePin(GPIOB, GPIO_PIN_7, 0); /*turn the led on*/ //Pb7GPIO
106        //HAL_Delay(1000);
107        //HAL_GPIO_WritePin(GPIOB, GPIO_PIN_7, 1); /*turn the led off*/
108        // HAL_Delay(1000);
109
110
111
112
113    /* USER CODE END WHILE */
114
115    /* USER CODE BEGIN 3 */
116
117    /* USER CODE END 3 */
118

```

Figure 39: Code to Control Voltage on Front-End

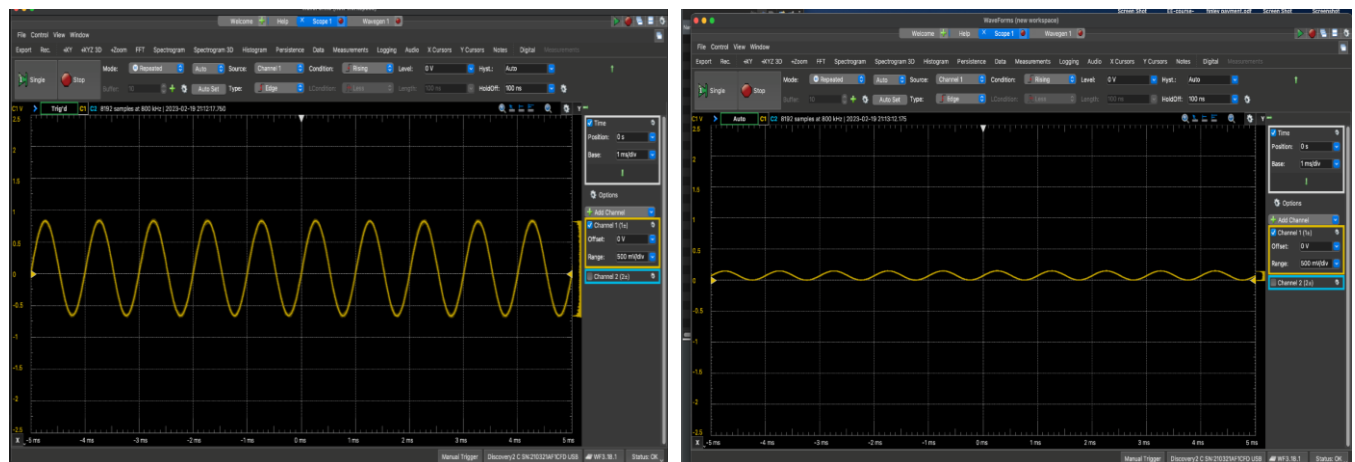


Figure 40: When the Code assign turn on/off input Voltage

7.1.3 Experiment 3: USB continue

In this experiment, the MCU team was testing how to use the STM32 and turn it into a USB that can transfer the data. The experiment was dependent on the coding in IDE and a simple circuit that is connected to the STM32. MCU team got the simple circuit soldering for the USB that can connect directly to the STM32 without using a breadboard. After connecting the new circuit to the computer and running the code for STM32, Hercules SETUP utility can receive the message successfully.

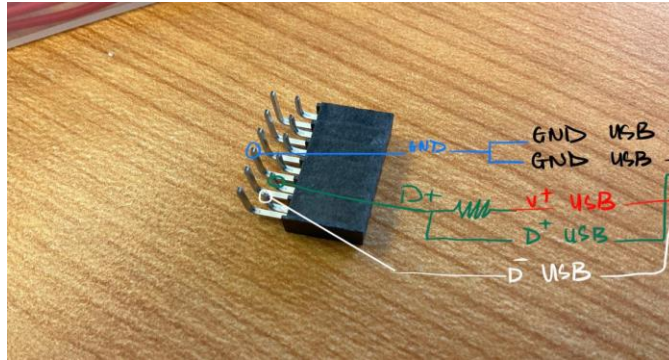


Figure 41: Pin and USB Cable Circuit

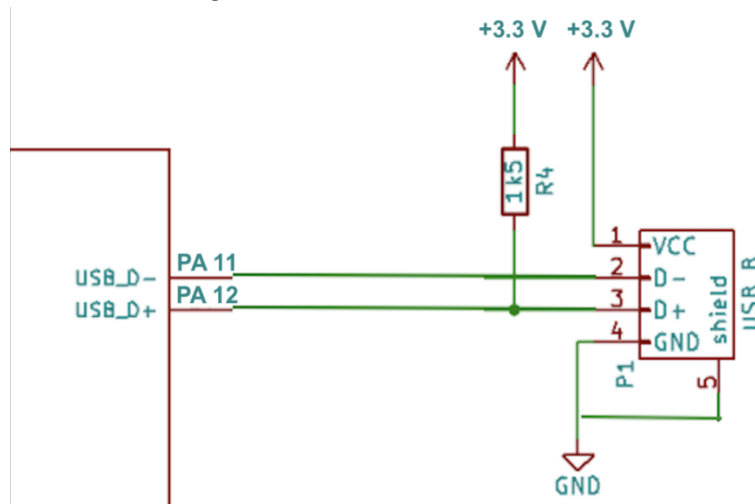


Figure 42: Schematic for USB



Figure 43: USB Cable Circuit after Soldering

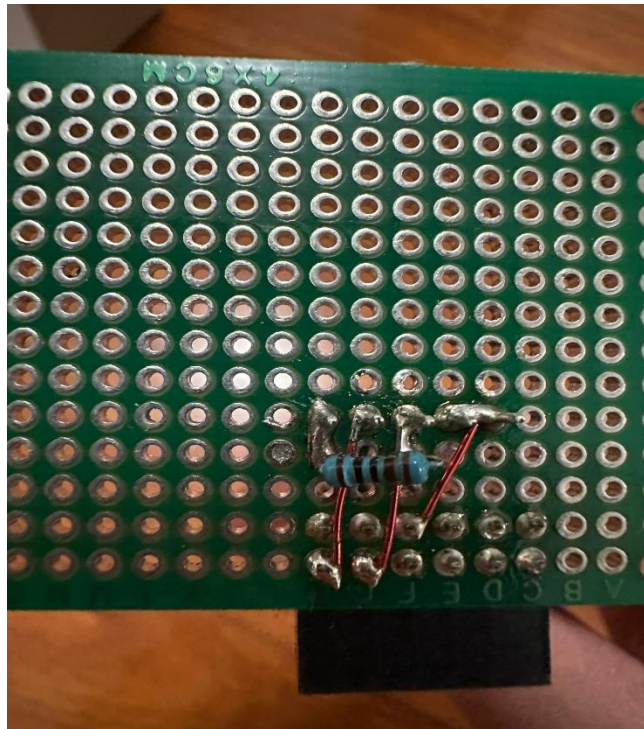


Figure 44: Close up of the Soldering Circuit

7.1.4 Experiment 4: MCU Testing with Front-end

MCU team is successfully able to use the GPIO on the STM32, the code is modified to use the buttons on the breadboard in order to control the 3.2V DC voltage input for the Front-End. STM32 was connected directly to the PCB of the Front-End for DC, preamplifier, and attenuator testing. The testing was successful. MCU is able to control the PCB with the button circuit.

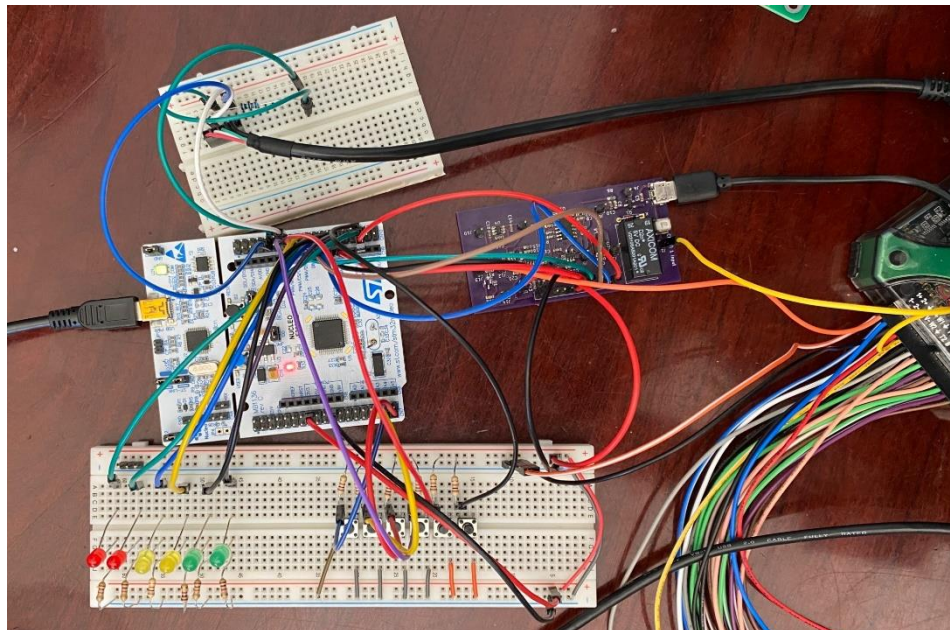


Figure 45: Testing between Front-end and MCU

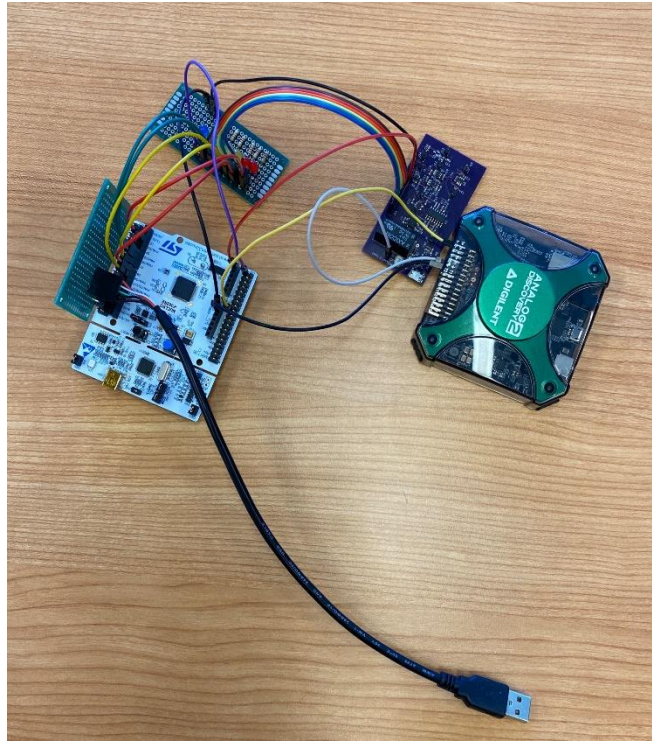


Figure 46: Connection between Front-end and MCU with Soldering Circuits

Connection					
Pin Name	Purpose (Hardware)	Purpose (Software)	Pin Name	Purpose (Hardware)	Purpose (Software)
PA0		ADC1_In	PA12		USB DP
PA8 (1/0)	Amplifier 1:10 (on/off)	GPIO_Out	PB4 (1/0)	Amplifier 1:1 (on/off)	GPIO_Out
PA9 (1/0)	AC/DC and AC	GPIO_Out	PB10 (1/0)	Amplifier 1:2.5 (on/off)	GPIO_Out
PA10 (1/0)	Attenuator (12.5/1.25)	GPIO_Out	PC7 (1/0)	Amplifier 1:5 (on/off)	GPIO_Out
PA11		USB DM			

Table 8: The Pin Connections of STM32 Board

In addition to transferring data from the STM32 to a PC using a Python script via USB, the MCU team also successfully resolved the testing for GPIO/PWM. The process for transferring data from the STM32 to the PC involved the MCU handling input signals via ADC and sending data to the PC through USB. The PC, in turn, recognizes the virtual com port of the MCU and receives the data. The MCU was also able to receive data from the PC via USB by using a Python script. This allowed the PC to send commands to the MCU, which could then change the MCU configuration accordingly.

7.1.5 Experiment 5: ADC Speed Testing

MCU successfully adjusted ADC sampling time and ADC clock. STM32F303RE default values, 72Mhz clock, 1.5 sample time.

```
void changeADC1Clock1() {
    HAL_ADC_Stop_DMA(&hadc1);
    LL_RCC_SetADCClockSource(LL_RCC_ADC12_CLKSRC_PLL_DIV_1);
    HAL_ADC_Start_DMA(&hadc1, (uint32_t*)adc_buff, ADC_BUFF);
}
```

Figure 47: Code for Changing ADC1 Clock

```
void changeSampling0(ADC_HandleTypeDef* hadc) {
    HAL_ADC_Stop_DMA(&hadc1);
    LL_ADC_SetChannelSamplingTime(ADC1, LL_ADC_CHANNEL_1, LL_ADC_SAMPLINGTIME_1CYCLE_5);
    HAL_ADC_Start_DMA(&hadc1, (uint32_t*)adc_buff, ADC_BUFF);
}
```

Figure 48: Code for Changing Sampling Time

To change ADC clock sampling time, we need to write our own function and call functions in Lower Layer libraries such as stm32f3xx_ll_adc.h and stm32f3xx_ll_rcc.h.

7.2 Experimentation of GUI

We developed a GUI application in Python using PyQt5 library that reads serial data and displays it on a plot. The GUI allows the user to specify voltage time division, time scaling values and DC offset for the plot, after that user can run and stop reading the data from the USB port. Besides, the GUI also allows the user to communicate with MCU from the user's commands.

7.2.1 Experiment 1: Reading Data from The USB Port Testing

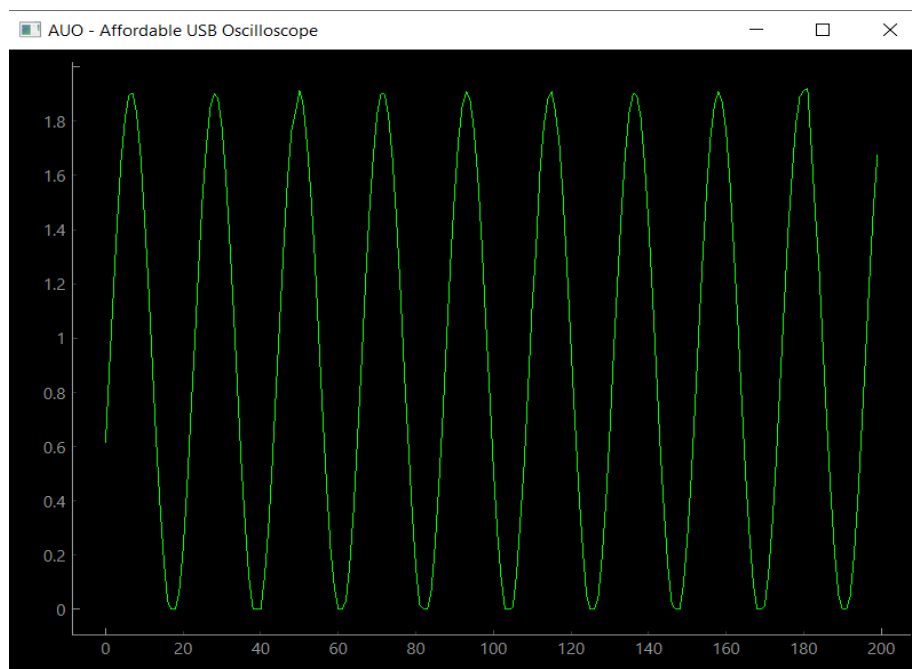


Figure 49: The Plot that Reading Data from USB Port

In this testing, we wrote code that used the PyQtGraph library to create a simple real-time plot of data received over a serial connection allowing the user to visualize and analyze the data in real time. The plot updates at a specified interval, and the data is converted to a voltage value before being plotted.

7.2.2 Experiment 2: Functions Testing

In this testing, we created a GUI that has various controls to adjust the voltage scale, time scale, and DC offset of the plot, as well as run and stop buttons for reading data from the USB port. The plot can update immediately when we change the value of frequency, amplitude, type wave... of the input signal. We also add the `plot_10240_points()` method that is used to plot the whole buffer of MCU in a single shot using a timer function. This method is called when the "Single(Plot 10240 points)" button is clicked. Moreover, we also add the cursor function into the GUI to measure certain parameters of a waveform.

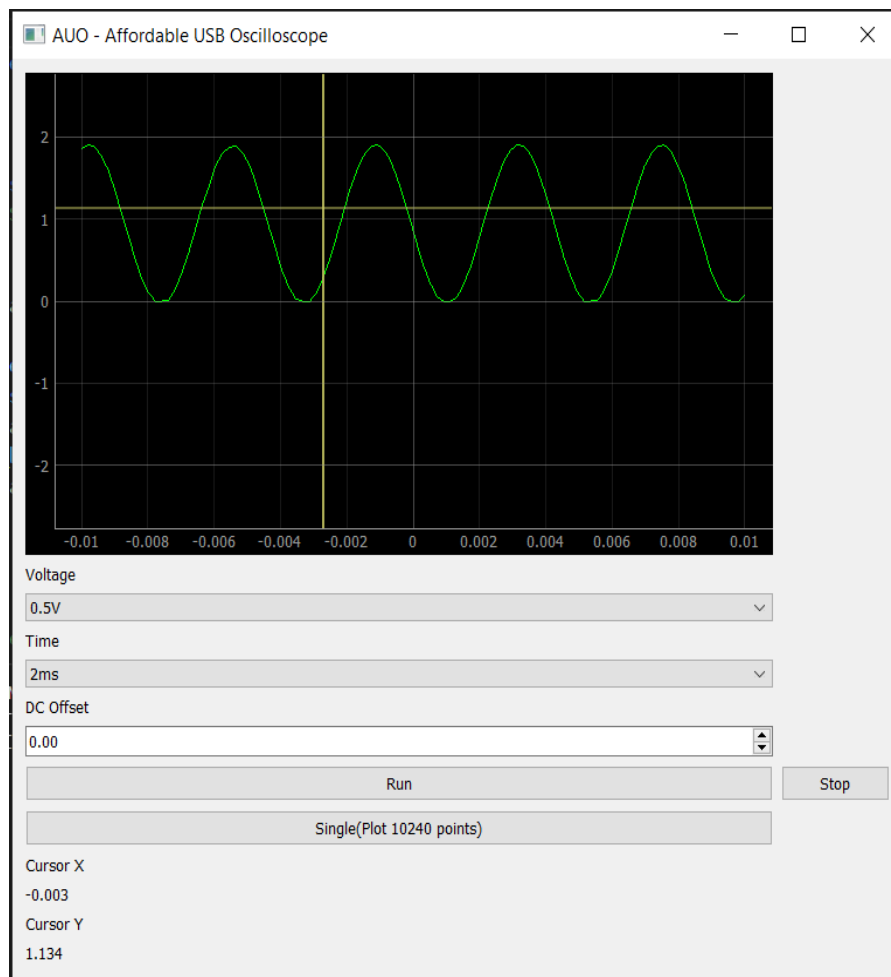


Figure 50: The GUI after adding some Functions and Buttons

7.2.3 Experiment 3: Communicate to MCU Testing

We added some functions like attenuator, AC/DC, amplifier, sampling time, ADC clock into this GUI to communicate to MCU. By using these functions, the user can control the Front-End voltage and convert analog to digital speed of the MCU.

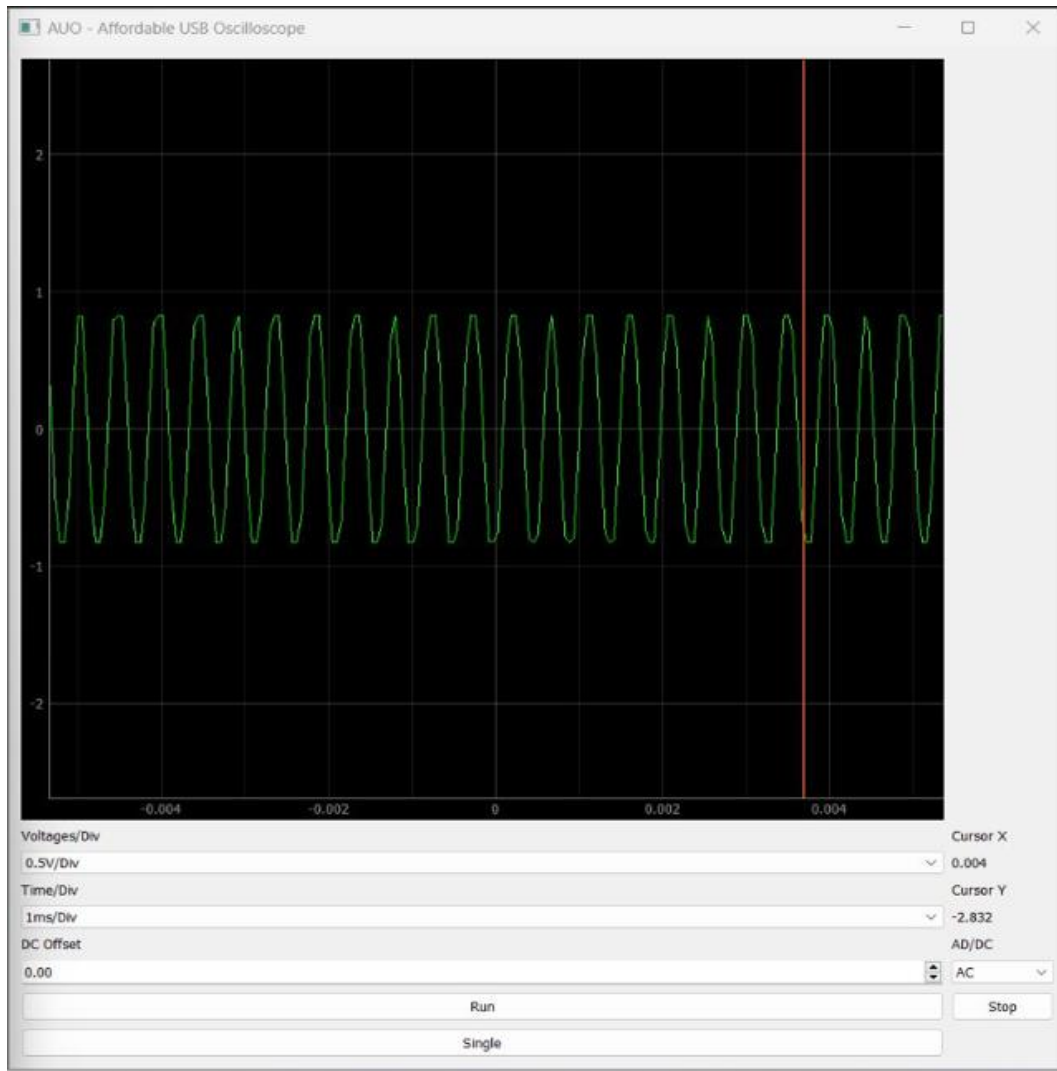


Figure 51: The Final GUI

7.3 Front-End/MCU/GUI Experimentation

7.3.1 The Whole Project Testing

To obtain a correction factor for calibration purposes, a difference in amplitude versus frequency test is performed to create a correction table. The following procedure outlines the steps to conduct this test:

- Step 1: Connect the input of the Front-End design to the waveform generator of the Analog Discovery 2 device (AD2).
- Step 2: Connect the output of the Front-End design to the development board STM32 Nucleo-64 based on the table provided.
- Step 3: Connect the STM32 Nucleo-64 to the PC using a USB cable.
- Step 4: Run the microcontroller for the PC to recognize the USB cable.
- Step 5: Run the GUI Python script.

- Step 6: Open the Waveforms software provided by the Analog Discovery 2 device and configure the waveform generator to produce a sine wave with the desired amplitude and frequency.
- Step 7: Choose options for voltage/division and time/division on the user interface, then click "run" to display the signal.
- Step 8: Repeat steps 6 and 7 to measure other values based on table below and frequency from 1 -500kHz each step 50kHz

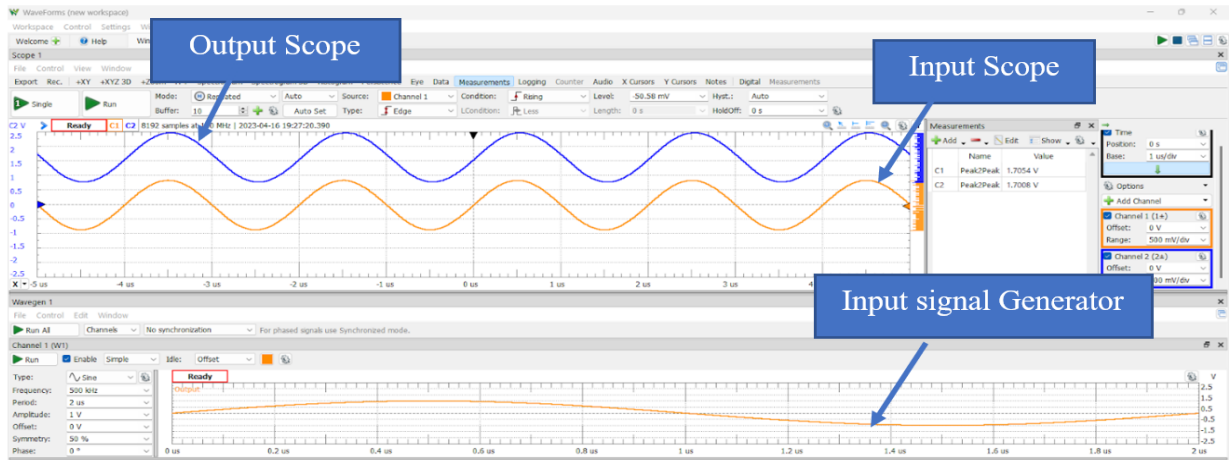


Figure 52: Testing Result by AD2

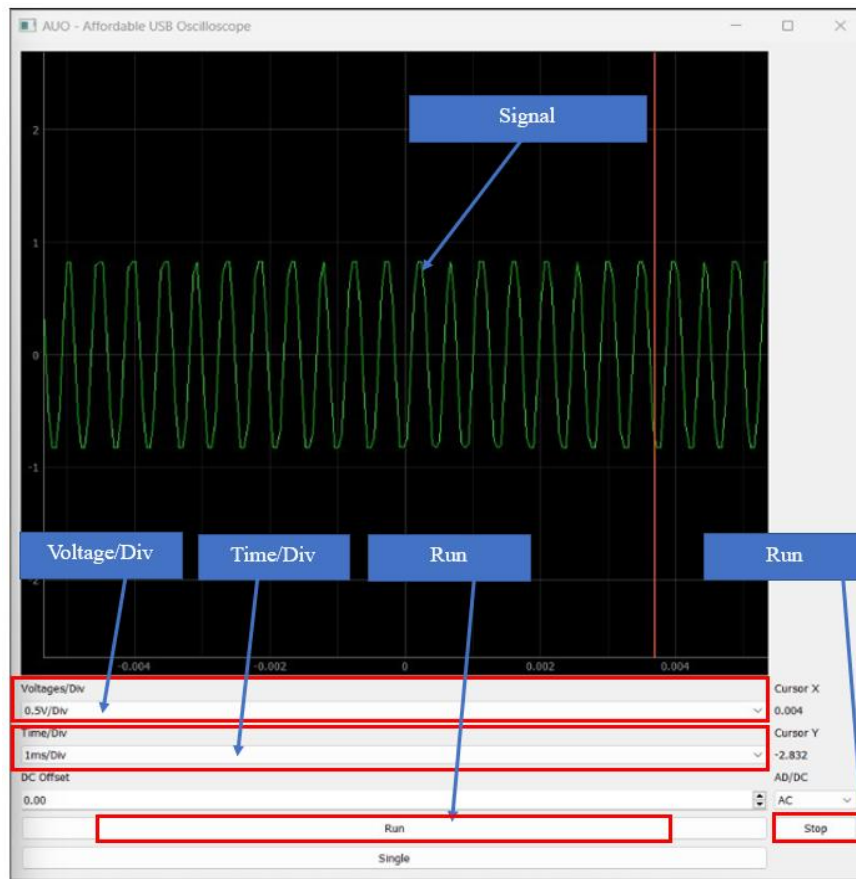


Figure 53: GUI Navigation

Attenuator	Gain	V _{input} (V)	V _{out} theoretical (V)	V _{out} experiment (V)	Error (%)
1.25 - 1	1 - 1	1V	0.8	0.794	0.76
	1 - 2.5	0.78	1.56	1.535	1.63
	1-5	0.378	1.52	1.47	3.40
	1 - 10	0.196	1.568	1.49	5.23
13 - 1	1 - 1	5	0.385	0.396	2.78
	1 - 2.5	5	0.962	1	3.80
	1-5	4	1.538	1.555	1.09
	1 - 10	4	3.077	3.124	1.50

Table 9: Correction Factor Testing Results

7.3.2 Frequency Response for Front-End PCB

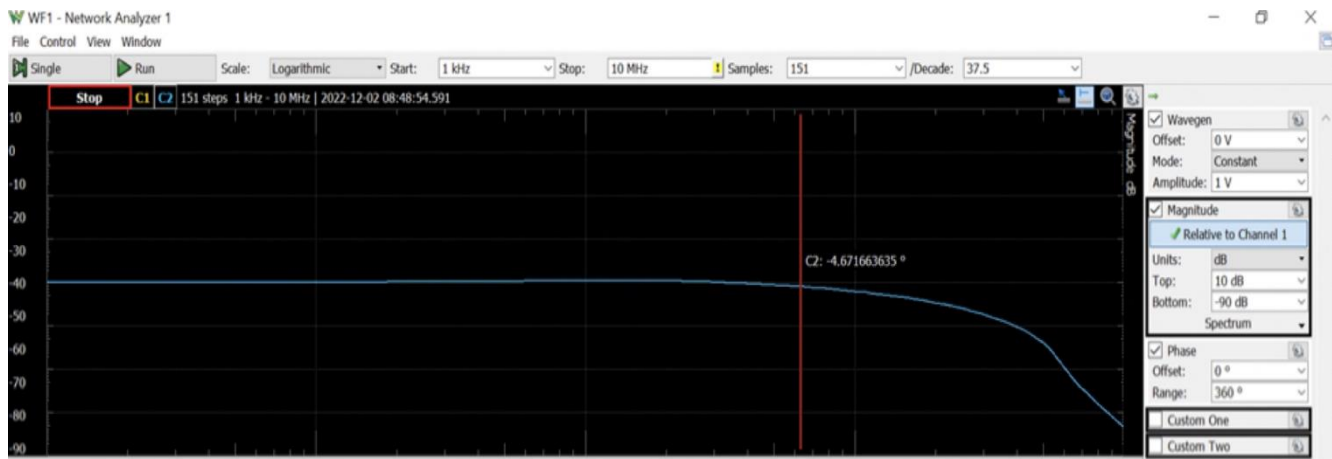


Figure 54: Network Analyzer by Analog Discovery 2

In Front-End Design, there are two attenuator paths and four amplifier paths, requiring total of 8 frequency response tests with different attenuation-to-amplification ratios of 1.25:1, 1.25:2.5, 1.25:5, 1.25:10, 13:1, 13:2.5, 13:5, and 13:10. To carry out these tests, follow these steps:

1. Connect the input of the Front-End circuit to the waveform generator of the Analog Discovery 2 device (AD2).
2. Connect the output of the Front-End circuit to the oscilloscope of AD2.
3. Open the Waveforms software provided by the Analog Discovery 2 device.
4. Configure the waveform generator to produce a sine wave with a starting frequency of 100 Hz and an amplitude of 1 V.
5. Configure the network analyzer to set the frequency range from 1 kHz to 10 MHz.
6. Click on the "Run" button in the Waveforms software to start the measurement.
7. The Waveforms software will automatically sweep the frequency range, measure the input and output signals, and calculate the frequency response of the circuit.
8. The frequency response data displayed a Bode plot

9. Repeat steps 3-8 for each of the remaining attenuator-amplifier paths.

The frequency response Bode plot of the Front-End circuit provides valuable insight into the behavior of the circuit with respect to different attenuation-to-amplifier ratios. This information can be utilized to optimize the design of the circuit for the intended application. Based on the Bode plot, it is observed that signals with attenuation-to-amplifier ratios of 1.25:1, 1.25:2.5, 13:1, 13:2.5, 13:5, and 13:10 can be transmitted up to 500 kHz. This indicates that the circuit can effectively amplify and transmit high-frequency signals with these ratios. However, signals with attenuation-to-amplifier ratios of 1.25:5 and 1.25:10 can be transmitted to 300 kHz

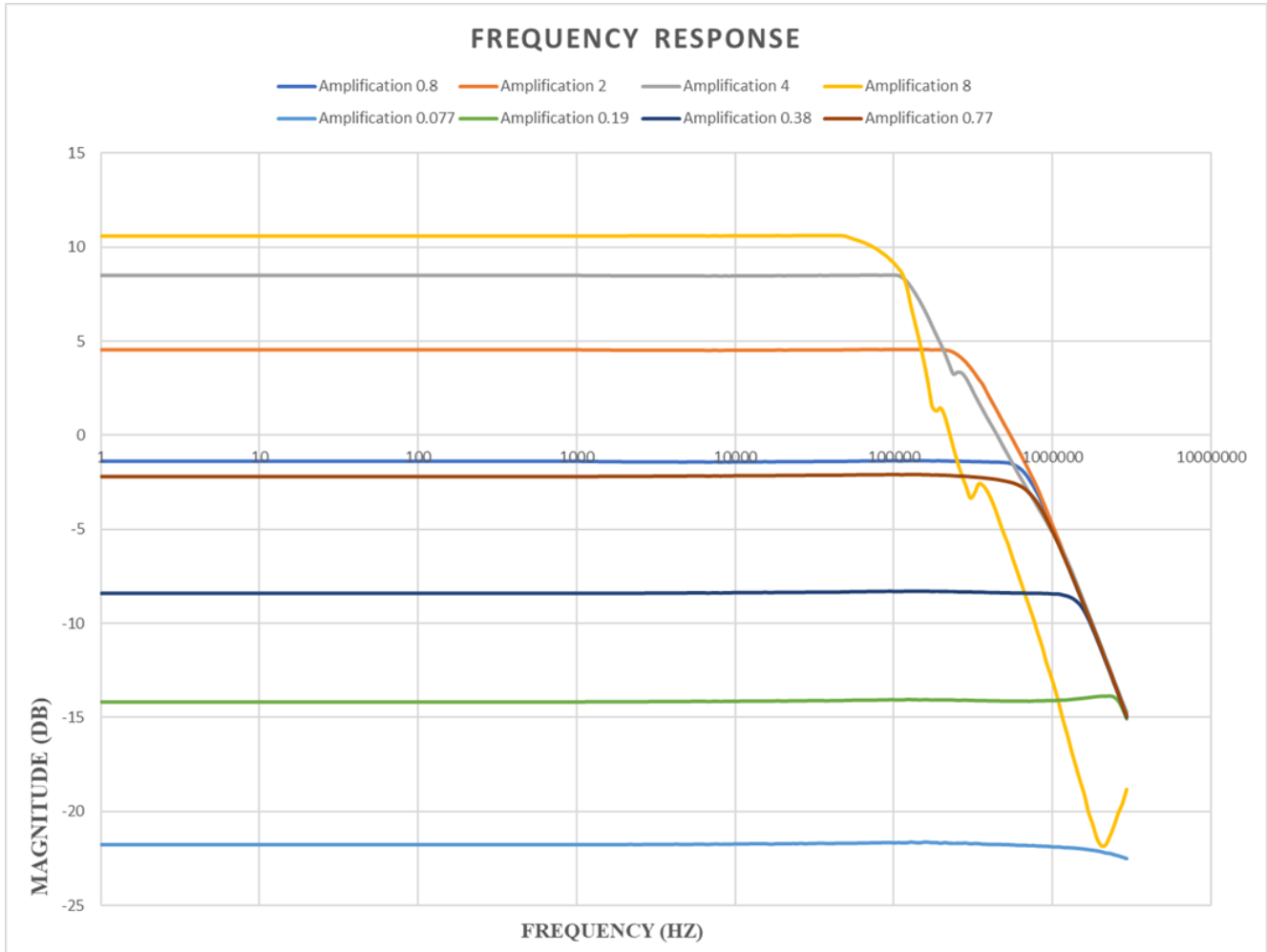


Figure 55: Frequency Response for Front-End PCB

7.4 Project Success Evaluation

7.4.1 Overall Project Evaluation

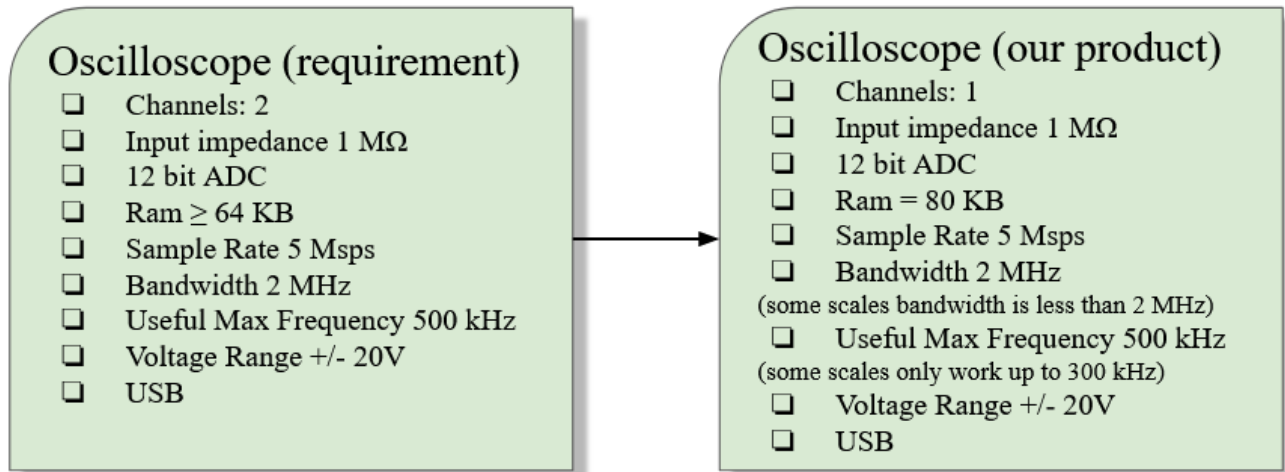


Figure 56: The Requirements at the beginning of the Project compares to our Product

The MCU team successfully managed to transfer data from the STM32 to a PC via a Python script using USB. To transfer data from the STM32 to the PC using USB, the MCU team had the MCU handle input signals via ADC and send data to the PC. The PC then recognized the virtual com port of the MCU and received the data. Additionally, the MCU team also succeeded in receiving data from the PC via USB by utilizing a Python script. With this set up, the PC was able to send commands to the MCU, which allowed the MCU to change its configuration accordingly.

7.4.2 Other Issues

- Reason for the project

At present, the AD2 and ADALM2000 cost approximately \$399 and \$252.79, respectively, which can be an overwhelming price for many students. As a result, to reduce the significant investment required for these devices, this project aims to develop a more affordable oscilloscope solution compared to the current market offerings. This new, cost-effective USB oscilloscope will fulfill the study requirements of undergraduate Electrical and Computer Engineering curriculum.

- Potential use of the project

The objective of this project is to design a new, affordable USB oscilloscope that must overcome three primary challenges. The first challenge is the hardware cost, which must remain low even if the chip is in a shortage crisis condition. The second challenge is performance, as the design must offer high accuracy. The final challenge is ensuring that the design incorporates the same features as other oscilloscopes in the market while also offering more features than other low-cost oscilloscopes.

The new affordable USB Oscilloscope will include a communication port by USB, 64GB RAM to transmit and receive data, 2 channels 12-bit ADC with 5Msps, at least 2Mhz clock speed, and enough general-purpose input and output to connect with the front end. This new device can also measure the scope of AC/DC signals.

- Alternatives to the implemented design that could impact costs and effects on resources use and the environment and community. Alternatives to the chosen design solution.
 - + Resistor ladder to reduce the number of digital switches and resistors in the front-end.
 - + Placing the MCU chip in the PCB instead of the plugin board to reduce the clutter and extra wires needed.
- Maintainability/maintenance of the final design solution
 - + Software: Everything on the GUI was run using Python with popular python libraries such as (Serial, Numpy, PyQt5, PyQtGraph) which can be updated without compromising the software side of the project.
 - + Hardware: if there were any defective parts, replacement is as simple as desoldering and resoldering that specific part.
- Retirement, replacement, or disposal of the project at the end of its “lifetime”

The STM32Nucleo board can be repurposed to different projects. Moreover, nontoxic parts such batteries were used so disposal of the device is simple.

8. Administrative Project Aspects

8.1 Project Progress

- **Front-End:** For the progress in Front-end, PCB design of Front-end was in the progress and was sent out to the PCB manufacturer in the first 2 months of Spring 2023. In addition, the -5V power supply was still needed to be tested on a breadboard PCB assembled for the single channel. -5 V power supply works fine in the PCB.
- **MCU:** To prepare for PCB-USB testing of the MCU, the first step is to configure the STM32 for USB data transfer. This involves coding in an IDE and connecting a simple circuit to the STM32. The MCU team has already soldered a simple circuit for USB, eliminating the need for a breadboard. With this setup, the MCU has successfully transferred data from the STM32 to a PC using a Python script, and vice versa. In addition to this, the MCU team has also made progress in testing GPIO/PWM. The team tested how to use the STM32 to control the 3.2V DC voltage input for Front-End, which was directly connected to the PCB for DC, preamplifier, and attenuator testing. Finally, the MCU team successfully adjusted the ADC sampling time and ADC clock using the default values of the STM32F303RE, which is a 72MHz clock with a 1.5 sample time.
- **GUI:** We developed a GUI application in Python using PyQt5. The GUI allows the user to interact with MCU to display and manipulate oscilloscope readings. The GUI settings allow the user to adjust the voltage and time per division, apply a DC offset, and control AC/DC settings. The voltage values displayed on the GUI are adjusted based on these settings, and the interaction between the GUI and the MCU involves scaling and inverting the voltage values to match the user-selected parameters.

8.2 Project Challenges

Front-End: Front-End hardware almost meets the project requirement, only some amplification path can work up to 300kHz. The amplifier with 22MHz Gain Bandwidth Product was used in the design.

The higher Gain Bandwidth Product should be used in order to let the front-end hardware can work with all the signals up to 500 kHz.

MCU: We complete most of the tasks. The team completed GPIO, USB (transfer/receive), Timers, DMA, and ADC. In the ADC task, the team is able to complete some frequency of input. We did not complete all frequencies from 1 to 500 kHz.

GUI: there are some improvements in the future: Automatic USB port detection, automatic timescale for the signal, and increasing the speed of data collection. Add trigger function and 2 channels function

– Changes to the design, tasks, schedule:

+ Front-End: did not change anything at this point yet.

+ MCU: did not change anything at this point yet.

+ GUI: At the beginning, we intended to develop a GUI based on the Waveforms Live application of Diligent but after that we changed to develop GUI using PyQtGraph because we lack knowledge on JSON encoding that is the communication protocol of Waveforms Live. PyQtGraph is a library of Python so we can easily get familiar with it.

8.3 Funds Spent

The following breakout cost is to design and test for a single channel, but the cost can be reduced if we buy a large number of components.

Cost for Single Channel Oscilloscope per unit	Cost per units (assuming 1000 units)
Components for single channel	\$20.082
PCB cost	\$13
STM32 Nucleo-64 Development Board	\$12.99
Miscellaneous Item	\$5
Total Cost (Single device)	\$51.072
All time Total (492/493)	\$599.49

Table 10: Cost Breakdown of Solution

8.4 Man-Hours Devoted To The Project

The table below summarized the total man-hours that were devoted to the project over the course of two semesters. It is broken down into each main technical project aspect.

Project Area	Approximate hour per week (hours)	Total hours (30 weeks)
Front-end/MCU	22	660
GUI	10	300
System Integration and Final Testing	10	60 (6 weeks)
Documentation and Course Delivery	2	60
Total (hours)		1080

Table 11: Total Man-Hours Devoted to The Project

8.5 Individual Team Member Contributions

- Front-End

- + Giang Nguyen: Worked on DC offset, buffer, LPF design, PCB design and testing.
- + Phu Duong: Worked on attenuator, AC/DC coupling, amplifier, voltage supplier +/-5V design, searched components, PCB soldering and testing.

- MCU

- + Bao Phan: Worked on USB transmit/receive, ADC converter, timer.
- + Thu Viet Minh Nguyen: Worked on GPIO for attenuator/preamplifier/DC, soldering USB-PCB.

- GUI

- + Duy Luong: Wrote control functions and GUI design.
- + Jasem A. J. M. Alsaedi: Wrote functions to receive and send live data to MCU and plot it continuously.

9. Lessons Learned

9.1 Additional Knowledge and Skills Learned

Front-End:

- 4 player PCB design skill.
- Practicing on using an oven to solder SMD components.
- Reading Technical Data sheet to understand in order to select the components which are suitable to our design.
- Problem solving: By working on hardware design projects that develop student's problem-solving skills and learn how to approach complex problems systematically.

MCU:

- Reading the data sheet (description of STM32F3 HAL and low-layer drivers, reference manual of STM32F3 family), understanding the concept and where to find information in it.
- STM32 Microcontroller software (STM32CubeIDE).
- Programming language in Python with multiple modules (sys, NumPy, serial, Python).

- Programming language in C (coding for STM32 microcontroller).
- Apply skills and knowledge from ECE 447 (microcontrollers class).
- Apply skills and knowledge from ECE 350 (Embedded Systems and Hardware Interfaces)

GUI:

- Programming language in Python with multiple modules (sys, NumPy, serial, Python).
- Programming language in Python using these libraries: PyQtGraph, PyQt5
- Understanding of Object-Oriented Programming concepts, including classes, objects, methods, and inheritance.

9.2 Teaming Experience

Throughout the project, our team had weekly meetings with the professor and team members to update on the progress. Through this project, everyone in the team learned valuable lessons about teamwork and the team environment. Since this was one of the longest team efforts for most of us, there were many situations that were a first-time experience for us. Some of these lessons learned are highlighted below.

9.2.1 Small Team

At the start of the project, our team immediately divided all members into three small teams, each with two members. The front-end team's goal was to ensure that the input going through the circuit produced the correct output. The microcontroller (MCU) team was focused on ensuring that the performance of the ADC was fast enough to avoid missing data while transmitting to the GUI. Meanwhile, the GUI team was responsible for receiving and transmitting data from and to the MCU, as well as performing live waveform generation. Each team worked separately on their respective tasks, with regular weekly meetings scheduled with our professor to provide updates on progress and answer any questions that arose.

9.2.2 Team Communication

Throughout the project, weekly meetings were mandatory for all members. This was the only efficient way that the team could exchange information about the project. During these meetings, we were able to have questions and concerns with our professor. For the last 2 months of the project, our team also increased the meeting time during the weekend to catch up with the project. Beside these weekly meetings, we also communicated online. Through online communication, we mainly update the team with all the due dates of the projects. We also use google doc and google slides for all the documents. By using these shared documents, all members can easily access the contents and edit.

9.2.3 Time tracking

Time tracking is crucial for the success of this project. Initially, during the first three weeks of the Fall 2022 semester, our group was a bit relaxed as we were unsure of the topic to choose for ECE 492. However, as soon as we finalized the project topic, the entire team worked relentlessly to conduct thorough research. We dedicated our personal time to delve into the assigned tasks and expand our knowledge base. The way we schedule our work was mainly based on the meeting with our professor each week. After each meeting, there would be some new tasks from our professor that we had to

complete before the next meeting. By doing so, it helped push our group to meet all the deadlines of the project. In ECE 492, our goal was to have the PCB done before ECE 493 so we could begin all the testing as the new semester started. However, due to the long holidays, we could not reach that goal which took us around 2 months total. 1 month in ECE 493 for PCB plus 1 month for soldering, testing, and validation. So, the first prototype of the PCB needs to be done before 493 to make time for testing, redesigning, and remaking the PCB.

10. References

- [1] Mercer, Doug. "Activity: Frequency Compensated Voltage Dividers, For ADALM1000." *Activity: Frequency Compensated Voltage Dividers, For ADALM1000 [Analog Devices Wiki]*, 22 Aug. 2022, <https://wiki.analog.com/university/courses/alm1k/circuits1/alm-cir-voltage-divider>.
- [2] "AC versus DC Coupling - What's the Difference?" *Siemens DISW*, <https://community.sw.siemens.com/s/article/ac-and-dc-coupling-what-s-the-difference>.
- [3] "Amplifier." *Wikipedia*, Wikimedia Foundation, 27 Nov. 2022, <https://en.wikipedia.org/wiki/Amplifier>.
- [4] "Buffer Amplifier." *Wikipedia*, Wikimedia Foundation, 6 Nov. 2022, https://en.wikipedia.org/wiki/Buffer_amplifier.
- [5] Colley, Stephen. "Pulse-Width Modulation (PWM) Timers in Microcontrollers - Technical Articles." *All About Circuits*, 7 Feb. 2020, <https://www.allaboutcircuits.com/technical-articles/introduction-to-microcontroller-timers-pwm-timers/>.
- [6] Contributor, TechTarget. "What Is Direct Memory Access (DMA)?: Definition from TechTarget." *WhatIs.com*, TechTarget, 21 Sept. 2005, <https://www.techtarget.com/whatis/definition/Direct-Memory-Access-DMA>.
- [7] Herres, David. "Understanding Sampling Modes in Digital Oscilloscopes." *Test Measurement Tips*, <https://www.testandmeasurementtips.com/understanding-sampling-modes-in-digital-oscilloscopes/>.
- [8] "STM32CubeIDE." *STMicroelectronics*, <https://www.st.com/en/development-tools/stm32cubeide.html>.
- [9] *STMicroelectronics*. https://www.st.com/resource/en/user_manual/um1725-description-of-stm32f4-hal-and-lowlayer-drivers-stmicroelectronics.pdf.
- [10] OzyOzk. "OzyOzk/Oguplot: Real Time Python Plot." *GitHub*, <https://github.com/OzyOzk/Oguplot>.
- [11] "PyQtGraph#." *PyQtGraph*, <https://pyqtgraph.readthedocs.io/en/latest/index.html>.
- [12] "Scientific Graphics and GUI Library for Python." *PyQtGraph*, <https://www.pyqtgraph.org/>.
- [13] Suyash458. "SUYASH458/SoftwareOscilloscope: A Software Oscilloscope for Arduino Made with Python and PyQtgraph." *GitHub*, <https://github.com/Suyash458/SoftwareOscilloscope>.

11. Appendix A: Proposal (ECE-492)



ECE 492 Senior Advance Design Project

Affordable USB Oscilloscope Project Proposal

Team members: Phu Duong
Duy Luong
Jasem A. J. M. Alsaei
Bao Phan
Giang Nguyen
Thu Viet Minh Nguyen

Faculty Advisor: Dr. Jens-Peter Kaps

ECE 492

Date of Submission: September 30th, 2022

Contents

1. Executive Summary	4
2. Problem Statement	4
2.1 Motivation and Identification of Need	4
2.2 Market Review	5
3. Approach	6
3.1 Problem Analysis	6
3.2 Our approach	6
3.2.1 Overview	6
3.2.2 Approach Hardware	6
3.2.3 Approach Microprocessor	6
3.2.4 Approach Graphical User Interface	7
3.3 Alternative Approaches	7
3.3.1 Plugin Board	7
3.3.2 Matplotlib	8
3.3.3 Alternative MCUs	9
3.4 Background Knowledge	9
3.4.1 Overview Oscilloscope Specification	9
3.4.2 Front-End Hardware Basic	10
3.4.3 Microcontroller (MCU)	11
3.4.4 Web Client & Graphical User Interface (GUI)	12
3.5 Project Requirements Specification	12
3.5.1 Mission Requirements	12
3.5.2 Operational Requirements	12
4. System Design	13
4.1 Functional Decomposition	13
4.2 Physical Architecture	16
4.3 System Architecture	16
5. Preliminary Experiment and Testing Plan	17
5.1 Preliminary Experiment	17
5.2 Testing Plan	17
5.2.1 Testing Plan for Individual Section	17
5.2.1.1 Analog Front-End	17

5.2.1.2	Microcontroller	18
5.2.1.3	Graphic User Interface	19
5.2.2	Testing Plan for The Project in ECE-493.....	19
6.	Preliminary Project Plan.....	20
6.1	Overview	20
6.2	Allocation of Responsibilities	21
6.3	Detail Project Plan Timeline ECE-492 (16 weeks)	21
6.4	Detail Project Plan Timeline ECE-493 (16 weeks)	22
7.	Potential Problems	22
7.1	Required Technical Skill Set.....	22
7.2	Analog Front-End	22
7.3	Microcontroller	23
7.4	Graphic User Interface	23
	References	24

1. Executive Summary

This project involves the design and construction of an affordable USB Oscilloscope device that meets study requirements of undergraduate Electrical and Computer Engineering (ECE) curriculum. The primary aim of the task is to design and construct an affordable USB Oscilloscope which will have a low cost even if the chip is in a shortage crisis condition. Secondly, the performance of the design must have a high accuracy. Finally, the design should have the same features as other oscilloscopes in the market such as AC/DC coupling, attenuator, amplifier, DC offset, buffer, low pass filter, and voltage protection. The device shall use the custom PCB design for the analog front end and microcontroller that have interface to the open source software (waveform live) via USB connection to display and navigate the results.

Initially in ECE 492, this project will be divided into three small separate teams such as front end, microcontroller, and GUI. Each team member will have their own task to do research and break down the problems to understand clearly their assigned part from the project. After coming up with the sketches and designs, the front-end, microcontroller, and GUI will be tested individually or in combination with other teams in the project throughout the period of ECE 492. For the front-end part, the input going through the circuit must have the correct output as we calculated by formulas. Next, the microcontroller (MCU) must perform ADC fast enough and not miss data while transmitting to GUI. Finally, the GUI must receive/transmit data from/to MCU and perform live waveform generation. After the individual testing period, each part will be fixed and improved to be more efficient and meet the main goal. At the very end of ECE 492, three groups will come together to start coming up with a full custom PCB design for the real affordable USB Oscilloscope device with the full connection of three parts. Moving forward to ECE 493, this will be the period of building and testing the device as a whole.

2. Problem Statement

2.1 Motivation and Identification of Need

For many years, undergraduates of the Electrical and Computer Engineering department have been necessitated to purchase the USB Laboratory Instruments such as the Analog Discovery 2 (AD2) or Advanced Active Learning Module (ADALM 2000). These USB Laboratory Instruments help students to learn and conduct lab experiments in their own time for their convenience without depending too much on the open hours of the lab rooms on campus. Thanks to the convenience of these devices, during COVID-19 period, students were able to use the Analog Discovery 2 and Advanced Active Learning Module to continue their studying at home safely without any obstacles. The pandemic has resulted in a high demand for these devices, as students around the country are trying to get a hand on it for their study. The surge in demand has challenged manufacturers around the world to quickly adapt and respond. Unfortunately, this outbreak has resulted in a serious chip shortage, which has led to a serious soaring of price for these devices.

Currently, the AD2 will cost around \$399 and the ADALM2000 will cost around \$252.79, which not every student can afford this overwhelming price for these devices. To reduce the significant amount of money that students need to invest for the equipment, this project will develop a less expensive oscilloscope solution compared to the current market. The new affordable USB Oscilloscope will include a communication port by USB, 64GB RAM to transmit and receive data, 2 channels 12-bit ADC with 5Msps, at least 2Mhz clock speed, and enough general-purpose input and output to connect with the front end. This new device can measure the scope of AC/DC signals with an average accuracy and precision measurement in order to meet study requirements of undergraduate Electrical and Computer Engineering curriculum.

2.2. Market Review





Model	Device Picture	Bandwidth (MHz)	Sampling Rate (MSa/s)	Number of Analog Channels	Sampling Resolution (bit)	Price (\$)
Analog Discovery 2 (AD2)		30	100	2	14	\$399
Advanced Active Learning Module (ADALM2000)		10	100	2	12	\$252.79
Hantek 2D72 - Handheld Oscilloscope with Digital Multimeter		20	250	2	8	\$174
FNIRSI-5012H 2.4-inch Screen Digital Handheld Pocket Oscilloscope		100	500	1	8	\$76.99

Table 1: Current Open Source Alternative Oscilloscopes

3. Approach

3.1 Problem Analysis

In order to solve the problems that we described above; the designed system has to meet these requirements: the device hardware should be low-cost while still fulfilling the requirements for most laboratory exercises in the undergraduate ECE curriculum. The total part cost should not exceed \$50. This is a tough challenge for our team to overcome because some electronic devices like chips, microcontrollers, boards... are in short supply and more expensive than before. Moreover, we have to make sure this laboratory instrument can measure the scope of AC/DC signals with an average accuracy and precision measurement. Therefore, in order to accomplish these requirements, we will be very selective with our design and component choices.

3.2 Our Approach

3.2.1 Overview

To solve the problem above, we design a device, and it must overcome three challenges. First, it is hardware cost, it must be low cost even if the chip is in a shortage crisis condition. The second is performance, the design must have high accuracy. Finally, the design should have the same features as other oscilloscopes in the market, and the design has more features than other low-cost oscilloscopes.

3.2.2 Approach Hardware

Before buying components, we must figure out what an oscilloscope can do, and what features it has. For example, AC/DC coupling, attenuator, amplifier, DC offset, buffer, low pass filter, and voltage protection. The ADC of the microcontroller accepts the input voltage for 0 to 3.3 Volt. Hence, the attenuators have 2 scales 1.25 and 12.5 for voltage (0.1 - 4) V_{pp} and (4 - 40) V_{pp} respectively. Therefore, one channel analog switch is needed to change the attenuator scale. The amplifier will have 4 scales 1, 2.5, 5, 1 and 10. A two channel analog switch will be used for switching the amplifier.

For the AC/DC coupling, we need a one channel analog switch to select AC/DC coupling. The buffer is needed for the current amplifier's purpose. The device will measure for the max frequency 500 kHz. Therefore, the Low Pass filter will help to eliminate all the signal noise that is higher than 500 kHz. The ADC only receives the positive voltage from 0 to 3.3 Volt, the DC offset with PWM will be used to adjust the signal to the right range of the ADC. After that, we design a circuit for each of them, and the next step is to figure out what component is a right fit for the circuit by applying calculation formulas and information from the datasheet of that component. Also, we need to keep the price as low as possible.

3.2.3 Approach Microprocessor

We have to consider what microprocessor is fit for our design. To get the right microprocessor, we must know what features are built in the microprocessor. For example, how much RAM does it have? How does a microcontroller transmit/receive data to/from a PC (USB/UART/Wi-Fi)? How many ADC bits and ADC channels does it have? What are the sampling rate and clock speed? The last consideration is the price and availability of a microcontroller in the market.

After reviewing 3 microcontrollers, such as the MSP430G2553 launchpad, STM32F302R8T6 Nucleo board, and STM32F303RET6 Nucleo board. We know that the first two do not meet requirements of the project. The RAM must be more than 64kB to store ADC data, and the microcontroller has 2 input channels. The first one, MSP430G2553 launchpad, has 512 RAM and 10-bit ADC. The second one, STM32F302R8T6 Nucleo board, has 16k RAM, 12-bit ADC, and 1 ADC channel. The last one is the STM32F303RET6 Nucleo board. It has 80k RAM, 4 ADC channels. It meets our project's requirements.

3.2.4 Approach Graphical User Interface

We will create a web-based GUI system that will act as a client to the web server running on our board. It is the primary interface between the user and the overall system. It will allow the user to enter the system configuration command and display a waveform data. The questions: how does the board connect to the PC? What is the script language to create the GUI system? Which application framework do we use? How does the GUI communicate to the board?

Our approach for GUI is that we will develop a web client that communicates with the web server running on our board in order to pass control and configuration information to the system and output waveform and status information. This web client will connect to WaveForms Live GUI via IP and port number. WaveForms Live is a data visualization tool that works with diligent products like the OpenScope MZ and the OpenLogger. Once a connection is made, we can control the WaveForms Live to launch its data visualization window.

3.3 Alternative Approaches

3.3.1 Plugin Board

This alternative approach comes from the chip shortage. Since the STM32F303RE single chip is hard to get, instead of disassembling the chip from the Nucleo board which has a chance of destroying the chip, another option is to design a plugin board that can connect directly to the Nucleo development board. This approach is similar to some of the Arduino shields that add additional functionality to the main board.



Figure 1: Arduino shield addon Board

This approach might introduce complexity to the overall design. It might also introduce more points of failure which makes it hard to figure out what's wrong.

3.3.2 Matplotlib

Another approach that might make it in the end is a Python extension that makes it easy to create plots and has a graphical user interface called Matplotlib.

A tale of 2 subplots

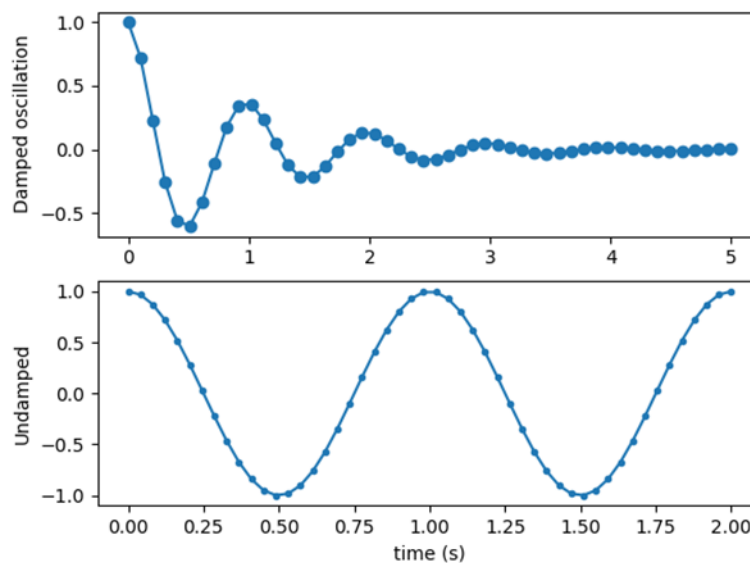


Figure 2: Matplotlib signal plot example

Matplotlib has an easy-to-use interface and a simple backend. Scripts can be used to collect the signal input data from the MCU and plot it into a graph with user controls. However, Matplotlib lacks

the professional and familiar look of Waveforms Live, and is generally slower in collecting data from the MCU and into the plot that is created to the end user.

3.3.3 Alternative MCUs

Currently, the top pick is the STM32F303RE MCU for its large flash/RAM capacity vs price. This project needs a large RAM capacity to store the signal data that is to be imported to the computer via USB. However, if the RE version were to not be available, the STM32F303RD can be used since it has the same RAM capacity but with reduced Flash capacity.

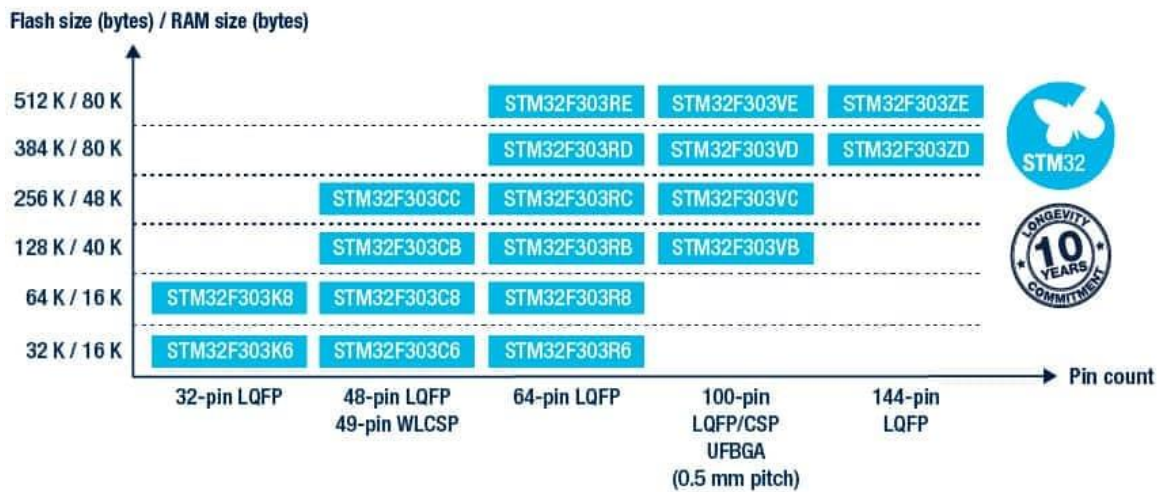


Figure 3: STM32F303 Series MCUs

3.4 Background Knowledge

3.4.1 Overview Oscilloscope Specification

- *Bandwidth:*

Bandwidth is a specified range of frequencies that can be accurately captured and measured by an oscilloscope. To ensure the probe's ability to capture signals, it is necessary to ensure that the bandwidth of the scope is higher than the operating frequency. The Rule of Five is often used as a rule of thumb. That means the bandwidth of the oscilloscope should be five times the highest frequency component in the signal. Using this rule, the error due to frequency limitation will be less than $\pm 2\%$. The oscilloscope with a bandwidth 2MHz will help to measure and analyze waveforms with a useful max frequency 500kHz.

- *Rise time:*

Rise time measures its ability to recognize and capture rising and falling signals. Rise time is a quantity closely related to bandwidth. Can be calculated as follows $\text{Rise Time} = 0.35/\text{Bandwidth}$.

- *Sample Rise:*

The sample rate is the number of samples that an oscilloscope can collect each second. You won't be able to accurately view your signal on the oscilloscope screen if your sample rate is inadequate and can be calculated as follows:

$$\text{Sample Rate} \geq 2.5 \times \text{frequency}.$$

- *The number of Channels:*

The oscilloscope is capable of reading multiple signals at once and displaying all of them simultaneously on one screen. This gives the ability to analyze and compare multiple signals at the same time. For the most widely used and popular two and four-channel oscilloscopes.

- *Vertical resolution:*

The ratio of the oscilloscope's maximum allowable input signal to the smallest detectable signal amplitude is known as the oscilloscope's vertical resolution. Resolution is often measured by the analog-to-digital converter's bit number (ADC). The number of bits multiplied by two determines the resolution. The resolution of a 12-bit converter is 4096.

3.4.3 Front-End Hardware Basic

- *Attenuation and Preamplifier:*

As a signal moves through a medium, attenuation occurs, which results in a decrease in the amplitude of the signal captured. In contrast to attenuation, the preamplifier is an amplifier that is used to boost the strength of input signals. When there isn't enough voltage for the load.

- *AC/DC coupling:*

The user can select the area of the signal she wishes to observe using an oscilloscope's DC (direct current) or AC (alternate current) coupling. DC coupling allows the whole signal (including DC and AC) entry, but AC coupling blocks the steady voltage.

- *DC offset:*

DC offset is an average amplitude deviation from zero. When the signal goes out of the range ADC of the oscilloscope, the DC offset adds the DC voltage to the incoming signal to move it in the range.

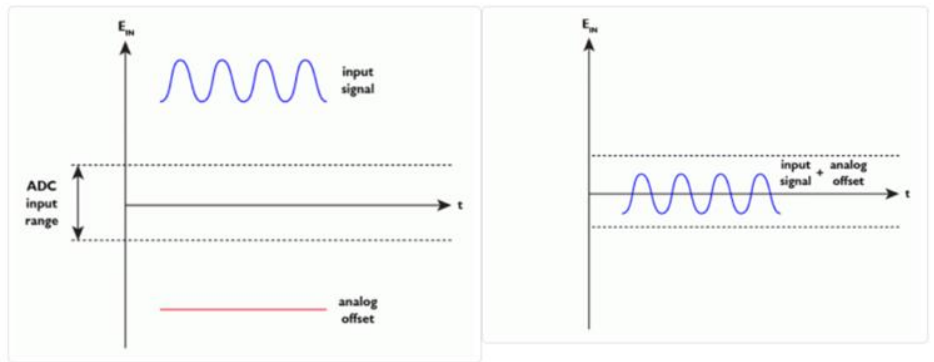


Figure1: Out-of-range signal

Figure 2: Signal brought into range by analog offset

Figure 4: Range signal by analog offset

- *Low pass filter*

The low pass filter (LPF) is a filter that allows the passing of the signal with low frequency and blocks the signal with high frequencies.

Low Pass Filter

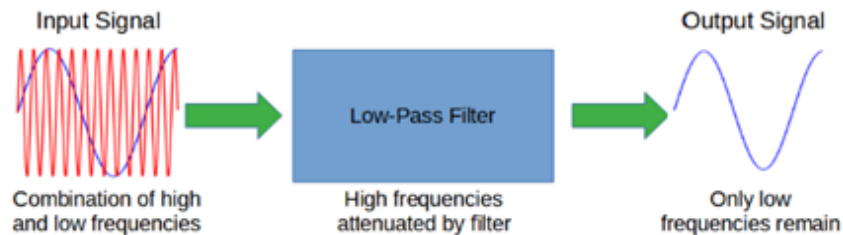


Figure 5: Low Pass Filter

3.4.4. Microcontroller (MCU)

The microcontroller is like a computer integrated on a chip and used to control electronic devices. It is a self-contained embedded system with peripherals, processors, and memory. The microcontroller is basically composed of the following parts:

- CPU- this is considered the brain of the microcontroller. It has the function of fetching and decoding instructions.
- Input/output port: used to control and communicate with devices such as monitors, LEDs, printers, etc.
- Memory: in microcontrollers memory is used to store data and programs.

- Serial port is the part that connects the microcontroller and other external peripherals
- ADC and DAC converters: ADC converters are used to convert analog signals to digital ones. In contrast, DAC is used to convert digital signals into analog forms.
- Timer and counter are a part of the clock speed of MCU, and it can be adjustable in configuration. It can be used as a delay or creating an event.

3.4.5 Web Client & Graphical User Interface (GUI)

A web client is a client-side application used for connecting to a web server over HTTP. It is typically a web browser or web app which displays web pages received from the server and allows users to interact with the web server. The web client and GUI will allow the communication between the user and the hardware simpler and efficiently without having to interface with the command line. In this project, we will develop a web client that communicates with the web server running on our board in order to pass control and configuration information to the system and output waveform and status information. This web client will connect to WaveForms Live GUI via IP and port number. WaveForms Live is a data visualization tool that works with Digilent products like the OpenScope MZ and the OpenLogger. WaveForms Live makes it easy to control and interact with instrumentation hardware by providing a cross platform graphical user interface that combines support for an oscilloscope, logic analyzer, GPIO, power supply and more into a single application.

3.5 Project Requirements Specification

3.5.1 Mission Requirements

The project shall develop an affordable USB-oscilloscope that has a low cost and satisfies the requirements for undergraduate students to do the Laboratory exercise at home. The device shall use the custom PCB design for the analog front end and microcontroller that have interface to the open source software (WaveForms Live) via USB connection to display and navigate the results.

3.5.2 Operational Requirements

- Input/output requirements
 - The device shall have two analog input channels and the output will display the signal on the computer screen via the WaveForms Live software. The users are able to change the configuration or setting that they want with the output signal.
- External Interface Requirements
 - The device will use the power source/ communication from the USB
 - The input impedance is 1 M Ω
- Function requirements
 - The device will use the microcontroller with analog front end to analyze the input signal as the following specification
 - The device has the working rang input from -20V to +20V

- The device will have the trigger option
- The signal bandwidth is 2 MHz
- The max frequency that that device can measure is 500KHz
- The device shall have the sample rate 5 Msps
- Technology and system-wide requirement
 - The analog front end with custom PCB design will be interface with microcontroller in order to process the input data
 - The microcontroller shall have at least 2 ADCs, RAM size is greater than 64K for fast respond
 - The device should be as small as possible
 - The device should be low-cost (less than \$50)
 - USB is using to data communication

4. System Design

4.1 System Functional Decomposition:

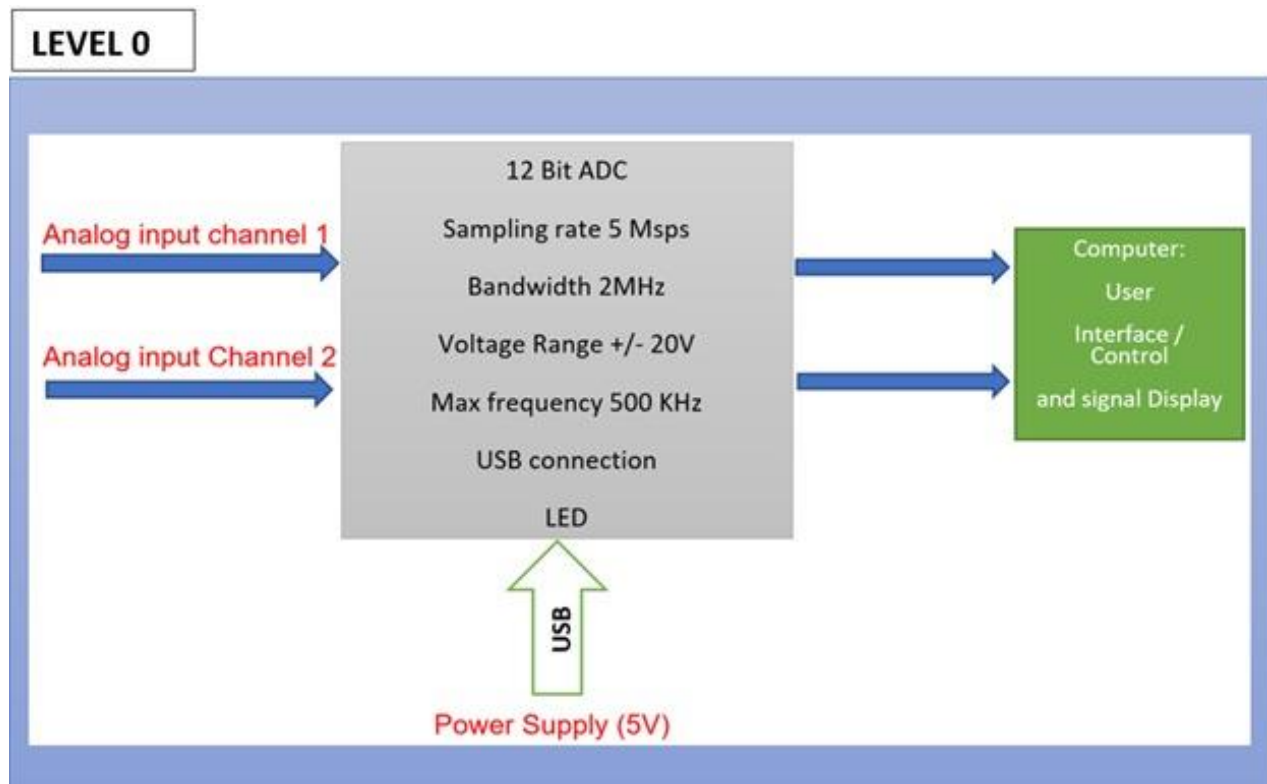


Figure 6: Level Zero Functional Architecture Block Diagram

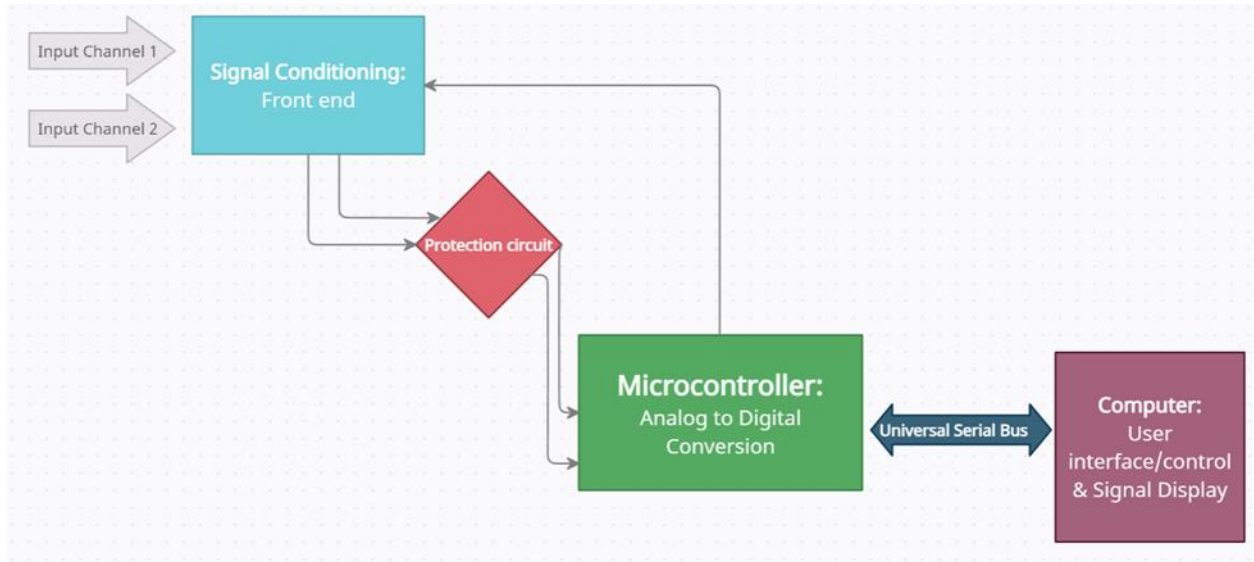


Figure 7: Level One Functional Architecture Block Diagram

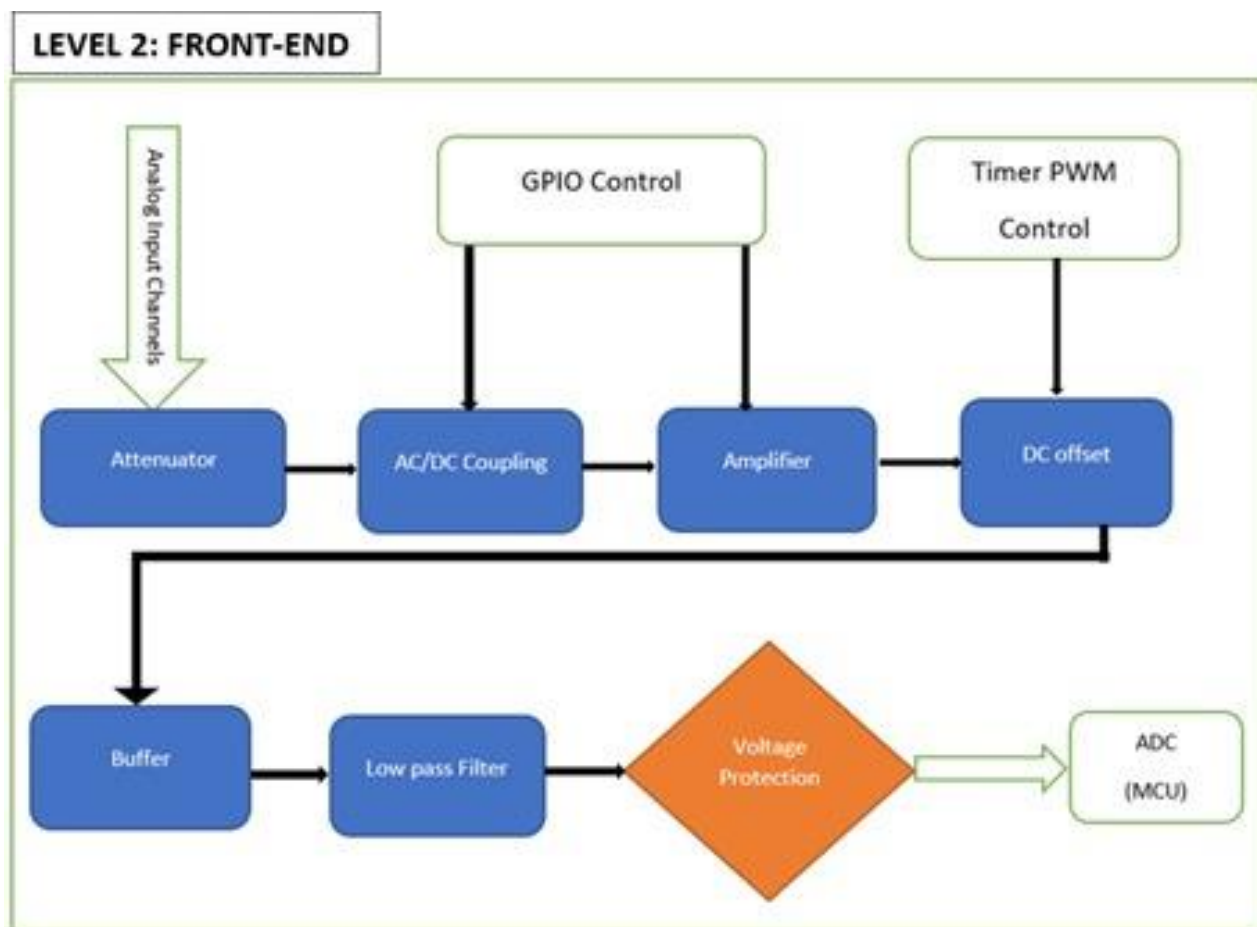


Figure 8: Level Two Front-End Architecture Block Diagram

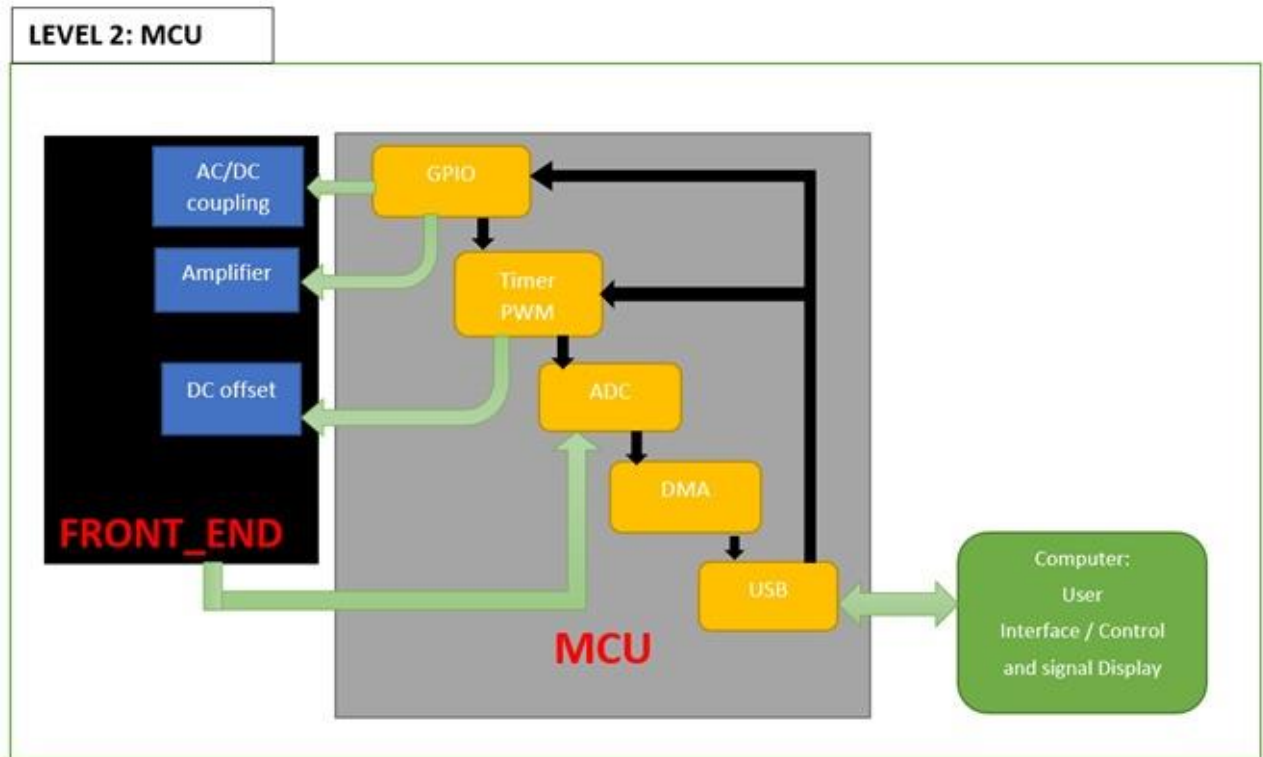


Figure 9: Level Two MCU Architecture Block Diagram

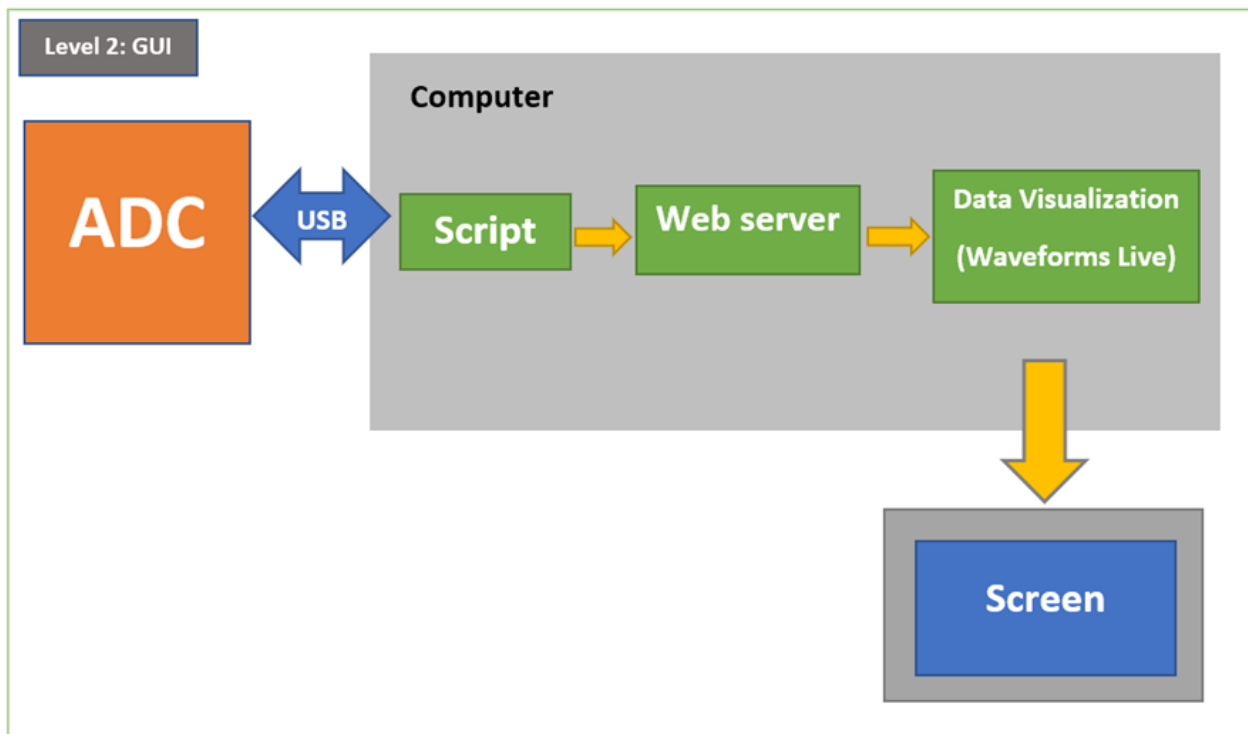


Figure 10: Level Two GUI Architecture Block Diagram

4.2 Physical Architecture

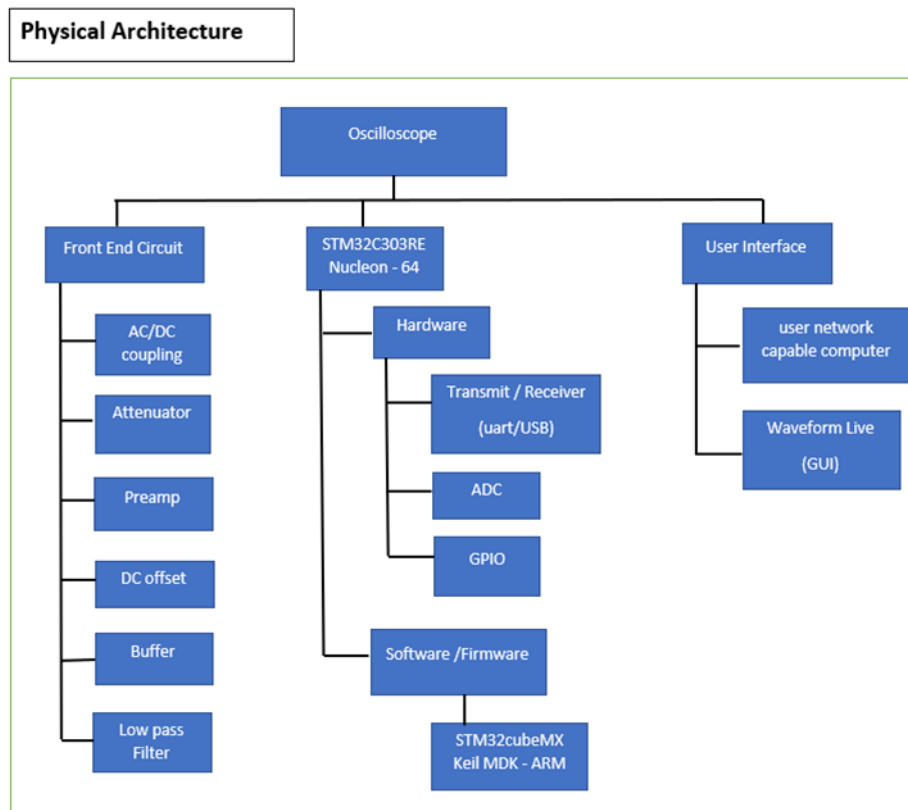


Figure 11: Physical Architecture

4.3 System Architecture

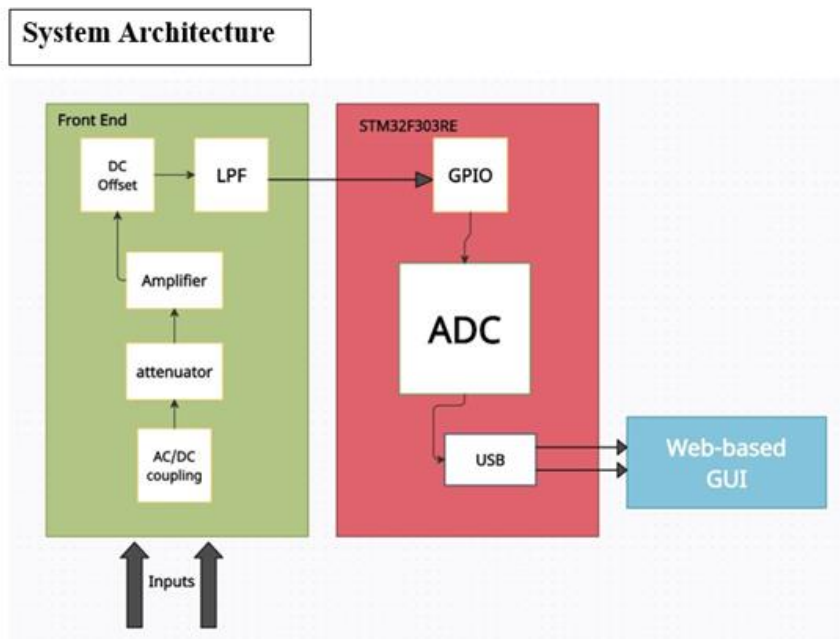


Figure 12: System Architecture

5. Preliminary Experiment and Testing Plan

5.1 Preliminary Experiment

To make the design work properly as we expected, we need to break the design down into 3 parts and test a single one of them. First is front-end part, the input going through circuit must have the correct output as we calculated by formulas. Next, the microcontroller (MCU) must perform ADC fast enough and not miss data while transmitting to GUI. Finally, the GUI must receive/transmit data from/to MCU and perform live waveform generation. A detailed testing list is described below and the testing list must be accomplished before performing final tests above.

5.2 Testing Plan

5.2.1 Testing Plan for Individual Section

5.2.1.1 Analog Front-End

- *Testing #1: AC/DC coupling + Attenuator*

The purpose of testing 1 is to make sure the AC/DC coupling function properly and the attenuator scale down the input voltage as designed. The AD2 or lab equipment will be used to send the input voltage then measure the output after attenuator AC/DC coupling.

- Testing for 1.25 scale attenuator. We supply the input signal (0.1- 4) Vpp then measure the out make sure the output is in (0.08 - 3.2) Vpp.
- Testing for 12.5 scale attenuator. The input signal (4 - 40) Vpp will be provided. Then checking the output make sure will be in (0.32 - 3.2) Vpp.
- AC/ DC coupling also will be toggled. we are able to see the signal shift to zero if the DC signal is filtered out.

- *Testing #2: Amplifier*

- Testing two will test the amplifier design to scale up the signal that meets the required input voltage of the ADC (0 - 3.3) Vpp. The AD2 or lab equipment will be used to send the input voltage then measure the output.
- The input signal from (0.08-3.2) Vpp will send to the amplifier then we toggle the analog switch to select the different amplifier scale (1, 2.5, 5, 10) then measure the output to make sure the scale multiply correctly.

- *Testing #3: DC offset*

- This test will test the Pulse Width Modulation (PWM) circuit design to ensure the DC offset fits the range of the ADC (0-3.3) V. The AD2 or lab equipment will be used to test the DC offset circuit.
- The input signal (0-3.3) Vpp will be supplied to our DC offset circuit. Then the DC voltage (0 - 1.65) V will be varied to offset the input signal the result will be seen on the computer screen.

- *Testing #4: Front-End assembly*

This final test for the front-end section will be conducted by different input voltages signal (0-40) Vpp, different frequency (0 - 500KHz), and different type of signal such as square, triangle, sinusoid signal within the range of the project requirement to ensure the output will be in the range of the selection ADC (0-3.3) V.

- Square signal: vary 2V each increment for Vpp and 100kHz each increment for the frequency then measure the output.
- Triangle signal: vary 2V each increment for Vpp and 100kHz each increment for the frequency then measure the output.
- Sinusoid signal: vary 2V each increment for Vpp and 100kHz each increment for the frequency then measure the output.

- *Testing #5: Spectrum Analyzer*

The goal of this test is to test the signal strength again frequency. Each channel will be tested separately. The input voltage from (0 - 40) Vpp (4 Vpp for each step test) from lab equipment or AD2 will be sent to the analog front end, and the frequency sweep from 10kHz to 5MHz to make sure the front end is met 2MHz bandwidth requirement.

5.2.1.2 Microcontroller

Before coming up with the full design for the microprocessor, the first step of the testing plan for the microcontroller is to be familiar with the microcontroller by using the STM32F303 Nucleo-64 board. By using this board and doing several experiments testing, we will have more idea on how the microcontroller works first hand. Then the main testing plan for microcontroller is to be familiar with the microcontroller library, functions, interaction frontend, and interact GUI (Pyserial).

- *Testing #1:* Switch pin High/Low (1/0) to control front-end component like VGA.
- *Testing #2:* Timer/PWM is used for input voltage in dc offset circuits. Therefore, the microcontroller can shift the input signal up and down. First, we use the STM32cubeIDE tool to set up configuration. Then, we use the HAL library and assign a compare and capture register (CCR) to control output voltage.
- *Testing #3:* ADC can convert analog to digital with a full sampling speed of the microcontroller. First, we use the STM32cubeIDE tool to set up a high-speed clock, ADC single-ended input mode, and DMA circular mode configuration. Then, we create a 16-bit array buffer and its size is 4096. Then, we use the HAL library to perform ADC and DMA functions. Finally, we display data on the PC terminal by USB/UART.
- *Testing #4:* Communication port USB and Pyserial - transfer/receive data to/from PC, programming language to interact MCU-PC. First, we use the STM32cubeIDE tool to set up USB configuration. Then, we create a message on the Nucleo board and send/receive the message to/from the PC terminal. On the PC terminal, we use Pyserial to communicate with the Nucleo

board. When we run both STM32cubeIDE and Pyserial at the same time, they can send/receive messages from each other.

- *Testing #5:* The final test for the microcontroller section is using AD2 to generate signal input then the MCU performs ADC. Also, we use direct memory acc (DMA) to transmit digital data to the PC by USB. From PC, we use Pyserial to receive/transmit data from/to MCU and plot the input.

5.2.1.3 Graphic User Interface

- *Testing #1:* Test specific Digilent Instrumentation Protocol functions.
- *Testing #2:* Test communication between web client and web server.
- *Testing #3:* Test the connection between the web client and WaveForms Live.
- *Testing #4:* Get a connection between a local host and “Waveforms Live” via JSON encoded command. JSON command set is used for its exclusive Digilent instrumentation protocol and creating legible commands that connect through an HTTP server.
- *Testing #5:* Making Python scripts that make live pseudo signals to plot via either Waveforms Live or Matplotlib to be used as an alternative approach.
- *Testing #6:* Creating scripts that capture live data from the USB interface and insert them into the local hosted server to be plotted by Waveforms Live.

5.2.2 Testing Plan for The Project in ECE-493

The purpose of this testing is to test the function and operation of our device.

- Testing #1: Front End - Microcontroller test (breadboard)
 - Function channel test.
 - Each channel will be tested separately. The goal for test one is to test the interaction of the microcontroller and the analog front end to make sure the coding interfacing to the front-end hardware and able to display the results by MATLAB or a small python script. The procedure of this test is to supply the input signal (0 - 40) Vpp (4 Vpp increasement each test). The microcontroller will toggle the analog switch to test the interaction with the microcontroller.
 - Same test will be conducted on the custom PCB.
- Testing #2: Calibration

The goal for this test is to find the correction factor of input signal in order to provide the correct input voltage for the ADC. The calibration procedure is to send the known input signal (0 - 40) Vpp, increase 2V each step, and record the amplitude of signal output. From the result, we will find the correction factor.

- Test #3: Microcontroller - Graphic User Interface

The purpose for this test is to test the interface of the micro controller and the GUI. The signal will be sent to the ADC of the microcontroller then the graph is plotted on the waveform live. The GUI

will change the command then we need to make sure that the MCU receives the command, perform the change, and then replot the graph.

- Testing #4: Whole assembly testing

This is the final test and the goal for this test is to make sure our device works properly and within the specification requirement above. The procedure for this test is to measure the different input signals from (0-40) Vpp, increment 4 V each step and make sure the plot is drawn correctly. In addition, the trigger function, DC offset function, and zooming function work shall be conducted.

6. Preliminary Project Plan

6.1 Overview

The project will be divided into two periods of time which will be done in ECE 492 and ECE 493. In the 16 weeks of ECE 492, the project will be divided by 3 sections, such as Analog Front End, Microcontroller, and GUI. The team members can be working parallel on development of each part then by the very late of ECE 492 timeline we will have demonstration of full design connected together.

For the analog front-end hardware design, the assigned team member will design the AC/DC coupling, attenuator, amplifier, DC offset, trigger, buffer, low pass filter and voltage protection by the project requirement specification. PSPICE will be used to simulate the design concept then the breadboarding process will be conducted. The testing procedure is using the AD2 to generate the signal for the hardware design then measure it back by the AD2 to make sure the hardware design meets the spec requirement. The PCB board will be drawn in KiCAD. Selecting operational amplifiers will be the key to keep the project at a low cost. Our planning shall use the analog switch to change the different scale of the attenuator and pre amplifier. The backup plan we will use the variable gain amplifier which can increase our cost a bit.

For the microcontroller section, we shall use the NUCLEO -F303RE development board to experiment with the microcontroller. STM32CubeIDE development platform will be used to generate, compile, and debug for STM32 microcontroller. We shall use the AD2 to generate the input signal for the development board, and the result after the ADC will be gone through the DAC of the microcontroller then will be displayed on the waveform application of AD2. The backup plan is to use STM32CubeMX for configuring the microcontroller and the KEIL software for programming. A small C++/ python program or MATLAB will be used to test the output signal of the microcontroller

For the GUI section, we shall build a server in order to create the IP address/Port to connect to the open source WaveForms Live. Alternative plan is to write a python script using together with the Simple Plotter library to draw the graph from the given digital data.

In the 16 weeks of ECE 493, our plan will finalize the design, continue testing, and do breadboarding the whole assembly for testing before making the custom PCB and debugging any error on any part of the project.

6.2 Allocation of Responsibilities

The individual tasks are divided based on the individual background, strength, skill sets and related project experience. Here is the breakdown task for each part of the project by team member:

- Phu Duong/ Giang Nguyen: Analog Front End circuit design and PCB.
- Bao Phan/Thu Viet Minh Nguyen: programming the Microcontroller to meet our task requirement.
- Jasem A J M Alsaedi/Duy Luong: GUI development on web server in order to display the output on WaveForms Live.

The weekly meeting will be hold to update the progress of the project and assign task for the following weeks.

6.3 Detail Project Plan Timeline ECE-492 (16 weeks)

- 1) Research the project (week 1- 4)
- 2) Do the proposal draft (week 5)
- 3) Proposal Presentation (week 6)
- 4) Development the detailed design and Testing (week 7-10)
 - + Week 7-8:
 - Test #1 and #2 - Front-end
 - Test #1 and #2 - Microcontroller
 - Test #1 - GUI
 - + Week 9-10:
 - Test #3 - Front-end
 - Test #3 - Microcontroller
 - Test #2 and #3 - GUI
- 5) Detailed design Presentation (week 11)
- 6) Testing parts (week 12-14)
 - + Week 12:
 - Test #4 - Front-end
 - Test #4 - Microcontroller
 - Test #4 - GUI
 - + Week 13-14:
 - Test #5 - Front-end
 - Test #5 - Microcontroller
 - Test #5 and #6 - GUI
- 7) Early prototyping and design document submission (week 14-16)

6.4 Detail Project Plan Timeline ECE-493 (16 weeks)

- 1) Front End-Microcontroller Development (week 1 - 5)
 - + Week 1: Testing #1: breadboard
 - + Week 2-3: PCB Design

- + Week 4-5: PCB assembly for front end and Microcontroller and redo test #1
- 2) GUI Development (week 1 - 5)
- 3) Testing
 - + Testing #2 (week 6)
 - + Testing #3 (week 4 - 6)
 - + Testing #4 (week 7 - 11)
- 4) Demonstration (week 12)
- 5) Report (week 13)

7. Potential Problems

7.1 Required Technical Skill Set

Limitations of technical skills and knowledge can become a huge problem during the project. There are multiple technical skills and areas of knowledge needed to get success for this project. The following areas are particular importance for our designed system:

- + PCB design
- + Programming language C, Python, Debug skill, reading datasheet and user manual, device drivers
- + GUI design using Python, JSON command, Matplotlib.
- + Postman application.

7.2 Analog Front-End

The success of an analog design relies on the designer's skill, knowledge, and ability to select the optimal set of parameters. In order to focus on a reliable solution, Spice simulations are performed and re-run, and the results are assessed over and over again. Moreover, analog designers have little choice but to accept the reality that Spice simulators are not as dependable or robust as their digital counterparts.

In addition, the complexity of the design will be a huge problem for the designer to understand the circuit. Choosing the right component from plenty of items which are available from the internet is time consuming for the designers too. Although individual functions are designed, the designers will face a lot of problems when all the individual functions are connected together. Last but not least, the noise of analog signals is the real problem that analog designers need to get rid of.

7.3 Microcontroller

During this project, one of the potential problems that might come up is the shortage of components to build the microcontroller unit. While building the MCU, the team might not be able to buy the correct chip due to the chip shortage in the market to build the MCU. Another problem can be sampling rate. The microcontroller must have 5Msps otherwise the collected data will not be accurate.

7.4 Graphic User Interface

We might not be able to connect the web client to WaveForms Live GUI due to the knowledge gap about the WaveForms Live. The WaveForms Live is written in a mixture of HTML, CSS, and Typescript. Therefore, it will be difficult for us to effectively understand the surrounding code of the Waveforms Live GUI, especially since no one in our team is really good at coding and computer science.

References

- [1] “OpenScope MZ: Open-source All-in-one Instrumentation,” Digilent. [Online]. Available: <https://store.digilentinc.com/openscope-mz-open-source-allin-one-instrumentation/>. [Accessed: 12-Sep-2022].
- [2] “WaveForms Live,” Digilent. [Online]. Available: <https://digilent.com/reference/software/waveforms-live/start>. [Accessed: 12-Sep-2022].
- [3] “What is a Web Client,” PhoenixNap. [Online]. Available: <https://phoenixnap.com/glossary/web-client>. [Accessed: 12-Sep-2022].
- [4] T. Jack, “How to Overcome Analog Design Challenges,” [Online]. Available: <https://www.talent-101.com/blog/how-to-overcome-analog-design-challenges#:~:text=Gain%2C%20bandwidth%2C%20signal%20distortion%2C,set%20of%20parameters%20to%20optimize>. [Accessed: 16-Sep-2022].
- [5] “STM32F303—Mixed-signal Microcontrollers (MCU), Arm Cortex-M4, 72 MHz CPU - STMicroelectronics,” [Online]. Available: <https://www.st.com/en/microcontrollers-microprocessors/stm32f303.html>. [Accessed: 16-Sep-2022].
- [6] “Using analog offset to maximize oscilloscope resolution,” [Online]. Available: <https://www.picotech.com/library/application-note/using-analog-offset-to-maximize-oscilloscope-resolution>. [Accessed: 16-Sep-2022].
- [7] “Low Pass Filter—Electronics,” [Online]. Available: https://electronicsreference.com/analog/low_pass_filter/. [Accessed: 16-Sep-2022].
- [8] Trompert, J, “How to overcome analog design challenges,” [Online]. Available: <https://www.talent-101.com/blog/how-to-overcome-analog-design-challenges>. [Accessed: 16-Sep-2022].

12. Appendix B: Design Document (ECE-492)



ECE 492 Senior Advance Design Project

Affordable USB Oscilloscope Design Document

Team members: Phu Duong
Duy Luong
Jasem A. J. M. Alsaei
Bao Phan
Giang Nguyen
Thu Viet Minh Nguyen

Faculty Advisor: Dr. Jens-Peter Kaps
ECE 492

Date of Submission: December 2, 2022

Table of Contents

1. Problem Statement	5
2. System Requirement Specifications	5
2.1 Mission Requirements	5
2.2 Operational Requirements	5
3. System Decomposition & Architecture	7
3.1 Level Zero Decomposition	7
3.2 Level One Decomposition	7
3.3 Level Two Decomposition	8
4. Background Knowledge	10
4.1 Analog Front-End	10
4.1.1 Attenuator	10
4.1.2 AC/DC coupling	11
4.1.3 Voltage Gain Amplifier	11
4.1.4 DC Offset using PWM Method	13
4.1.5 Unity Gain Buffer	15
4.1.6 RC Low Pass Filter	15
4.2 MCU	17
4.2.1 ADC Converters	17
4.2.2 DMA	17
4.2.3 STM32CubeIDE	17
4.2.4 HAL Driver	17
4.2.5 LL Driver	18
4.3 GUI	18
5. Detailed Design	19
5.1 Analog Front-End Schematics	19
5.1.1 Power Supply -5V Schematic	19
5.1.2 USB Soft Start Schematic	20
5.1.3 Attenuator Schematic	20
5.1.4 AC/ DC Coupling Schematic	21
5.1.5 Gain Amplifier Schematic	21
5.1.6 DC Offset using PWM, Buffer, and RC Low Pass Filter Schematic	22
5.1.7 The Whole Design Schematic	22

5.2 Analog Front-End Component Selection.....	23
5.2.1 Power Supply -5V	23
5.2.2 Attenuator.....	23
5.2.3 AC/ DC Coupling	24
5.2.4 Gain Amplifier.....	24
5.2.5 DC Offset using PWM.....	25
5.2.6 Buffer and RC Low Pass Filter.....	25
5.3 MCU	25
5.4 GUI Design	26
6. Prototyping & Early Testing Progress Report	28
6.1 Analog Front-End Testing	28
6.1.1 Attenuator Path (1.25:1) Simulation.....	28
6.1.2 Attenuator Path (12.5:1) Simulation.....	33
6.1.3 USB Soft Start Circuit.....	38
6.2 MCU Testing.....	38
6.2.1 Receive Data from ADC Python Code	38
6.2.2 Testing Variable ADC Sampling Rate.....	41
6.3 GUI Testing.....	46
6.3.1 Speed Testing for PyQtGraph.....	46
6.3.2 Plot Testing.....	47
7. Testing Plan for ECE493	47
7.1 Analog Front-End Testing	47
7.2 MCU	47
7.2.1 GPIO Testing.....	47
7.2.2 Front End - Microcontroller Testing (Breadboard)	48
7.2.3 PCB - GPIO/PWM Testing	48
7.2.4 PCB - ADC Function Testing.....	48
7.2.5 PCB - USB Testing (Transmit/Receive).....	48
7.3 GUI Testing.....	48
7.4 High-Level Overall System Testing.....	48
7.4.1 Calibration	48
7.4.2 Frequency Sweep.....	49
8. Task Allocations for Remainder of Project	49
8.1 Analog Front-End.....	49

8.2 MCU 49

8.3 GUI..... 49

9. Schedule for Remainder of Project 50

References 51

1. Problem Statement

For many years, undergraduates of the Electrical and Computer Engineering department have been required to purchase the USB Laboratory Instruments such as the Analog Discovery 2 (AD2) or Advanced Active Learning Module (ADALM 2000). These USB Laboratory Instruments help students to learn and conduct lab experiments in their own time for their convenience without depending too much on the open hours of the lab rooms on campus. Thanks to the convenience of these devices, during COVID-19 period, students were able to use the Analog Discovery 2 and Advanced Active Learning Module to continue their studying at home safely without any obstacles. The pandemic has resulted in a high demand for these devices, as students around the country are trying to get a hand on it for their study. The surge in demand has challenged manufacturers around the world to quickly adapt and respond. Unfortunately, this outbreak has resulted in a serious chip shortage, which has led to a serious soaring of price for these devices.

Currently, the AD2 will cost around \$279 and the ADALM2000 will cost around \$210, which not every student can afford this overwhelming price for these devices. To reduce the significant amount of money that students need to invest for the equipment, this project will develop a less expensive oscilloscope solution compared to the current market. The new affordable USB Oscilloscope will include a communication port by 2.0 USB with speed 12 Mbit/s, 64KB RAM to transmit and receive data, 2 12-bit ADCs with 5Msps, at least 2 MHz bandwidth, and enough general-purpose input and output to connect with the front end. This new device can measure AC/DC signals up to 500 KHz in order to meet study requirements of undergraduate Electrical and Computer Engineering curriculum.

2. System Requirement Specifications

2.1 Mission Requirements

This project involves the design and construction of an affordable USB Oscilloscope device that meets study requirements of undergraduate Electrical and Computer Engineering (ECE) curriculum. The primary aim of the task is to design and construct an affordable USB Oscilloscope which will have a low cost. Secondly, within an affordable price, any student can purchase and play around with the device. Finally, the design should have the same features as other oscilloscopes in the market such as AC/DC coupling, attenuator, amplifier, DC offset, buffer, low pass filter, and voltage protection. The device shall use the custom PCB design for the analog front end and microcontroller that have interface to the open source software via USB connection to display and navigate the results.

2.2 Operational Requirements

- Input/output requirements
 - The device shall have two analog input channels and the output will display the signal on the LCD via the open source software. The users are able to change the configuration or setting that they want with the output signal.
 - The device has the working range input from -20V to +20V

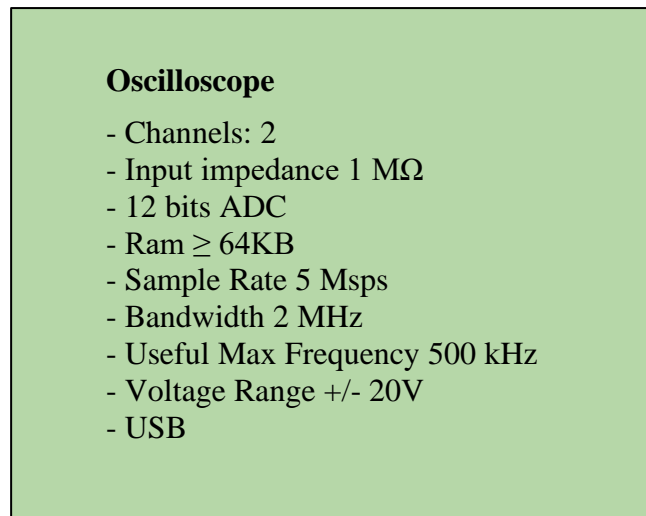


Figure 1: Operational Requirements of the Project

- The input impedance is 1 MOhm
- The signal bandwidth is 2 MHz
- External Interface Requirements
 - The device will use the power source/ communication from the USB
- Function requirements
 - The device will use the microcontroller with analog front end to analyze the input signal as the following specification
 - The device will have the trigger option (GPIO trigger/Level trigger)
 - The max frequency that device can measure is 500KHz
 - The device will have a sample rate 5 MS/s
 - The device will have a DC offset.
 - Memory for channels is 40KB, each channel is 20KB
- Software requirements
 - The software application can run on Linux, Window, OSX
 - It should run on any platform which supports the following packages: python, Matlab, NumPy, C, C+, ...
 - Speed minimum is 10 fps
 - Easy to use and operation
- Technology and system-wide requirement
 - The analog front end with custom PCB design will be interface with microcontroller in order to process the input data
 - The microcontroller shall have at least 2 ADCs, RAM size is greater than 64K for fast respond
 - The device should be as small as possible

- The device should be low-cost (less than \$50)
- USB is using to data communication

3. System Decomposition & Architecture

3.1 Level Zero Decomposition

The Level zero provides the overview of the top down oscilloscope design. The system is shown as in figure below which include 2 analog input channels, user command, DC power supply from USB. The output is the waveform signal which will be shown in the GUI.

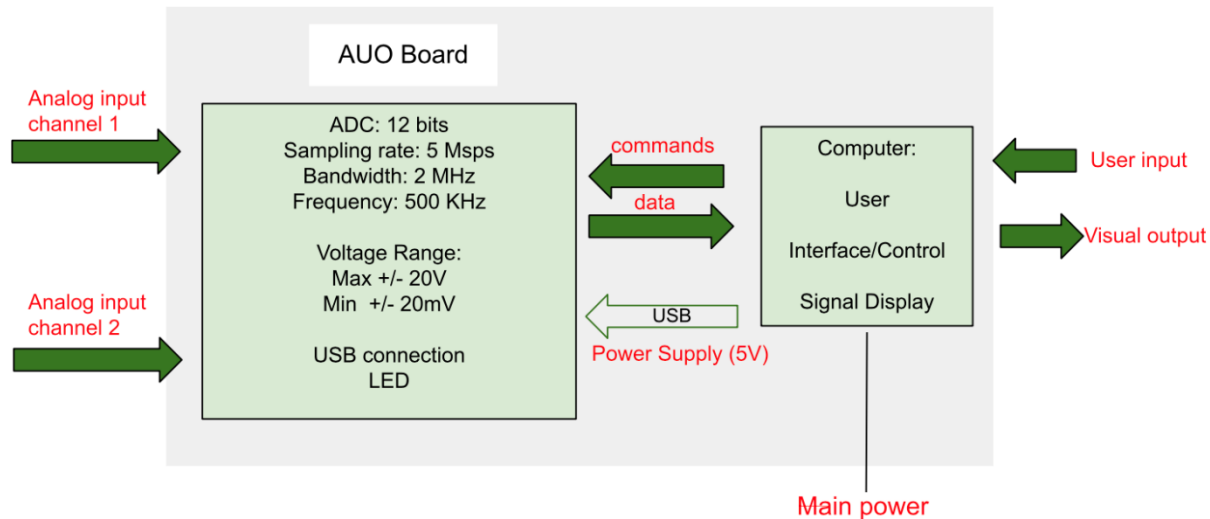


Figure 2: Level Zero Functional Architecture Block Diagram

3.2 Level One Decomposition

The design is in the detail as the figure below, signal will be conditioning by the Front End then the microcontroller which has an ADC function will convert/ process the analog signal to digital signal. By the communication through the USB, the data is transmitted to the GUI then the waveform will be plotted to visualize the analog signal on the monitor.

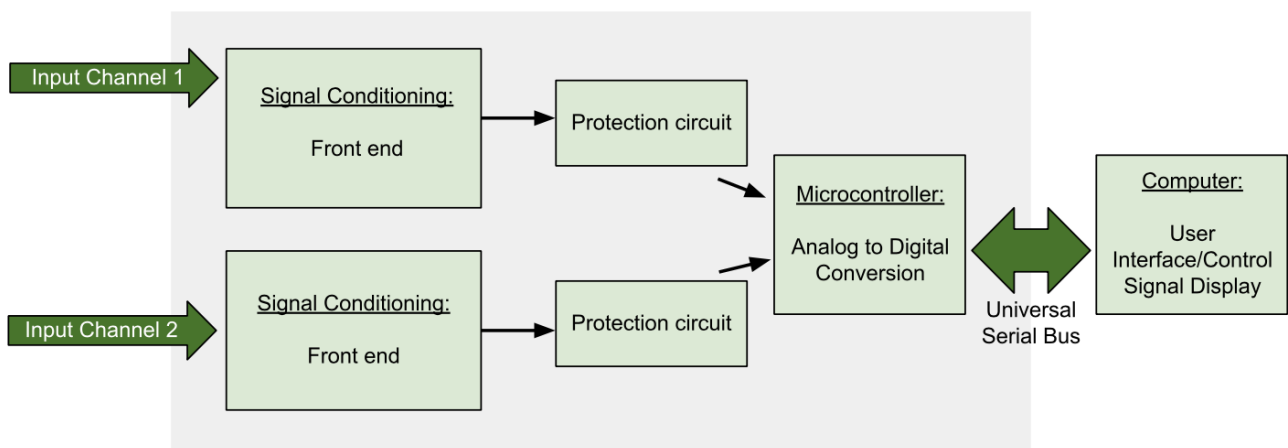


Figure 3: Level One Functional Architecture Block Diagram

3.3 Level Two Decomposition

Level 2: Front-End

The main hardware for the oscilloscope is the analog front end that is starting with the attenuator which is used to scale down the input signal. The DC/AC coupling function is used to block/unblock the DC offset that we want in the signal. The amplifier is used to scale up the signal to the range that the ADC can be read. Different scales will be controlled by analog switches by GPIO signal from the microcontroller. In addition, the signal will be offset above zero for the ADC is able to read. to strengthen the current, the buffer is added to provide the needed current to send to the ADC. The low pass filter will help to filter out the noise signal that is not needed. and finally, the voltage protection is used to protect the unwanted high voltage that suddenly appear to damage the microcontroller.

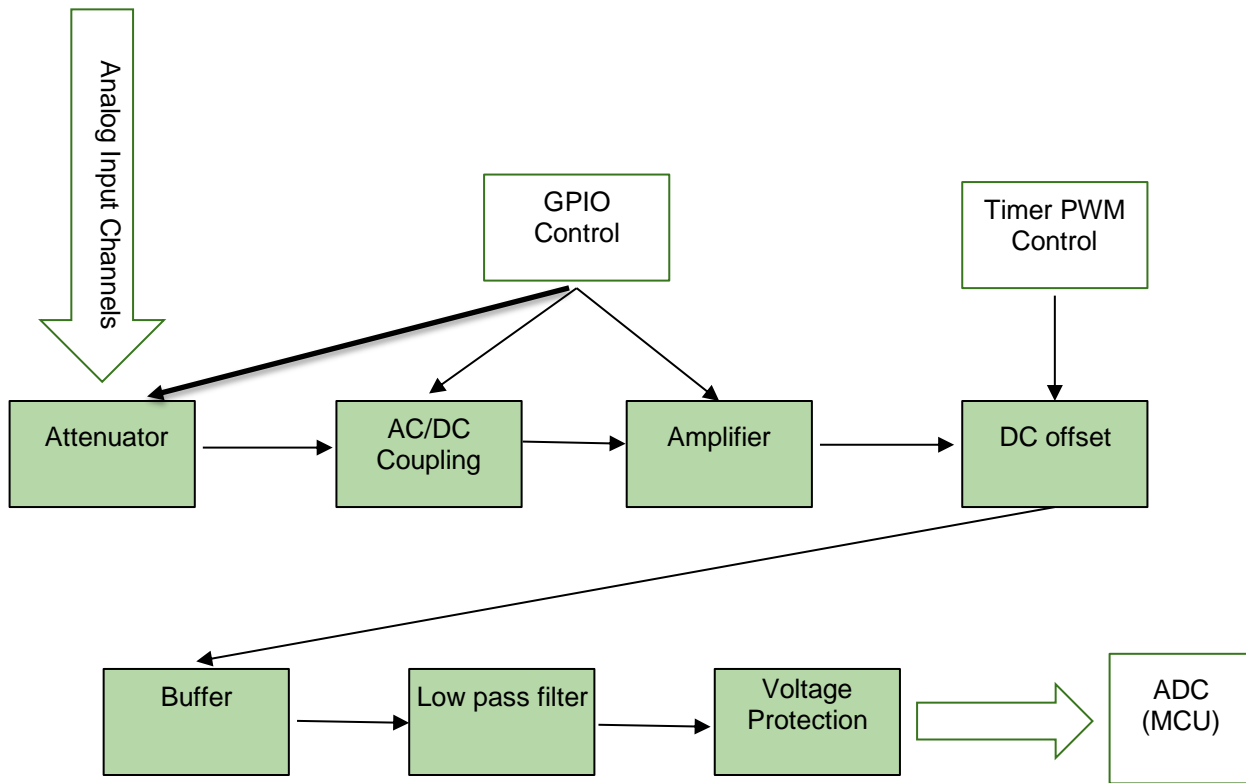


Figure 4: Level Two Functional Architecture Block Diagram - Analog Input Signal Preconditioning

Level 2: MCU

The MCU will receive the signal from the front-end and transfer it to the computer (GUI). The MCU includes GPIO, Timer PWM, ADC, DMA, and the USB. The GPIO will control the AC/DC coupling and amplifier of the front-end. The GPIO will choose specific options in AC/DC coupling and amplifier circuits to process signals. Also, the timer PWM will control the DC offset of the front-end by variable output voltage to shift signal up and down. The signal received from the front-end will go to the ADC, then move forward to DMA and then the USB. From the USB, there will be 2 feedbacks to the Timer PWM and the GPIO. The MCU will use USB to transfer/receive data to/from the

GUI. In transferring, the MCU transfers signal under digital form to GUI. In receiving, the MCU receives a signal from the GUI to change the FE setup, adjust the signal before it goes through the MCU.

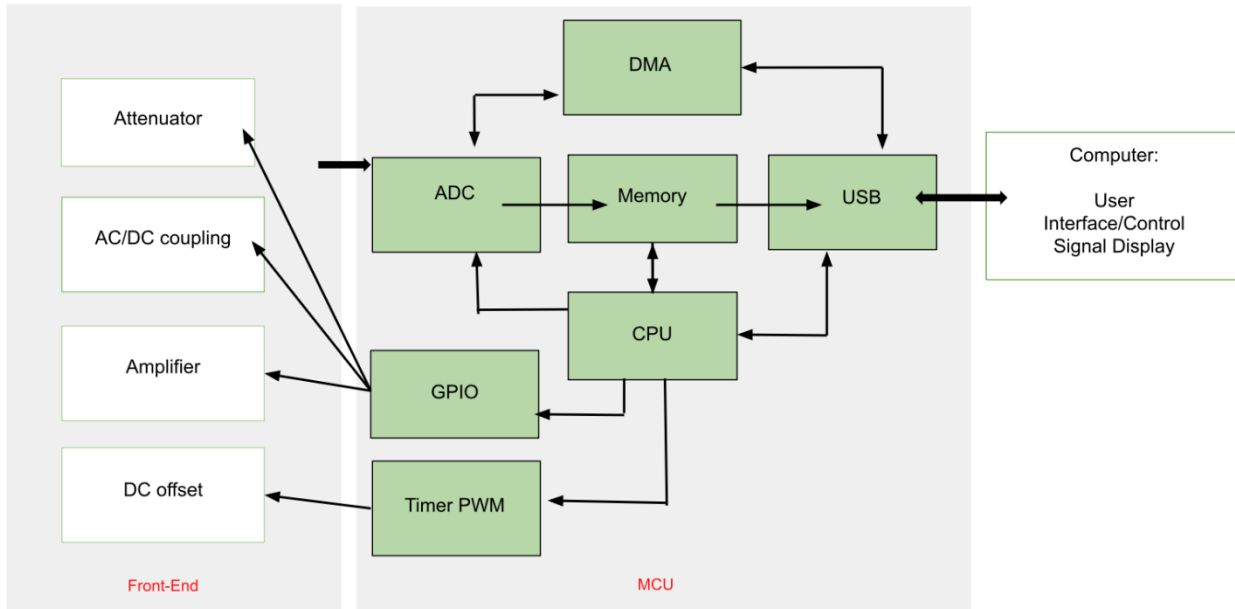


Figure 5: Level Two Functional Architecture Block Diagram - Microcontroller

Level 2: GUI

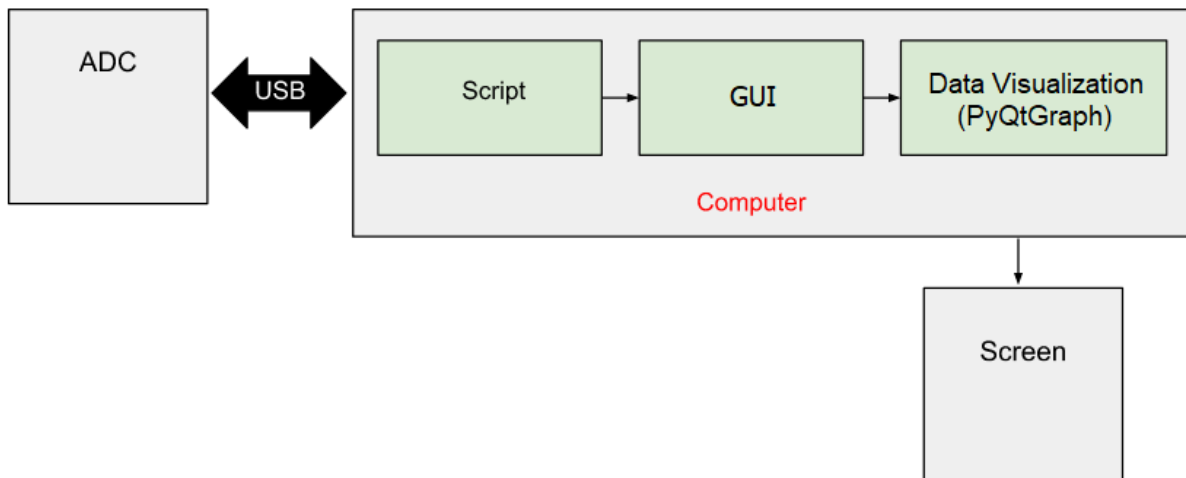


Figure 6: Level Two Functional Architecture Block Diagram - GUI

After the memory dump through USB happens, the data will be moved from the ADC into the main local machine. This data is interpreted by the computer as a hexadecimal number which needs to be further processed before being plotted into the live plot. The conversion from hexadecimal to a voltage number is needed because the cursor function of the plot will show whatever row value is shown which is why having the voltage value is correct. Furthermore, the signal will be capped between 0 to 3.3v

because of the ADCs limitations, this means that the input signal will not be the same as the data out of the ADC which is where further processing is needed to return the signal back to its original form before plotting.

- User Interface

This stage of the system will be the final stage before viewing the actual plotted data. The data will be plotted using a Python3+ based library called PyQtGraph. This Python library has a fast-refreshing rate between data plots and can easily do the required 2 channel scope, and it is all run locally on the main machine. This stage will also be used to control the systems voltage division and voltage switch for the input which will allow the ADC to accept more than its limit of 0 to 3.3V; this is done by attenuating the signal on the input and compensating for that on the output to revert it to its original state.

4. Background Knowledge

4.1 Analog Front-End

The Analog Front End includes multiple sections in order to provide the correct input signal to the ADC of the MCU. There are attenuators, AC/DC coupling, gain amplifiers, low pass filter, and DC offset using PWM in our front-end design. The basic background knowledge and calculation are used to select the hardware components are listed a below.

4.1.1 Attenuator

The attenuator uses a simple resistor to do voltage divider. To prevent the capacitance effects at the high frequency, the capacitors are connected parallel to the resistor which needs to satisfy the equation below.

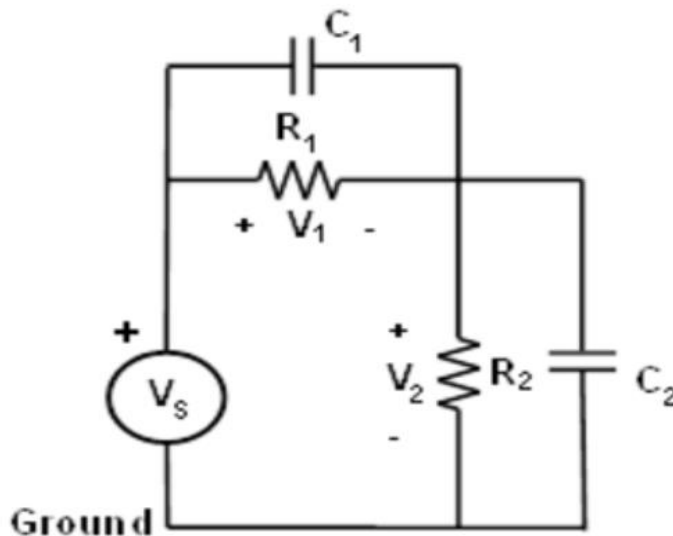


Figure 7: A frequency compensated voltage divider

$$R_1 * C_2 = R_2 * C_1$$

$$V_2 = V_S \frac{R_2}{R_1 + R_2}$$

Calculation for (1.25:1) Scale

$$V_2 = V_s \frac{R_2}{R_1 + R_2} = V_s \frac{780k}{780k + 200k} = V_s * 0.8$$

Select: $R_2 = 780 \text{ k}\Omega$

$$R_1 = 200 \text{ k}\Omega$$

$$C_1 = 7.8 \text{ }\mu\text{F}$$

$$C_2 = 2 \text{ }\mu\text{F}$$

Calculation for (12.5:1) Scale

$$V_2 = V_s \frac{R_2}{R_1 + R_2} = V_s \frac{80k}{920k + 80k} = V_s * 0.08$$

Select: $R_2 = 80 \text{ k}\Omega$

$$R_1 = 920 \text{ k}\Omega$$

$$C_1 = 0.8 \text{ }\mu\text{F}$$

$$C_2 = 9.2 \text{ }\mu\text{F}$$

4.1.2 AC/DC coupling

AC and DC stand for Alternating (Capacitive) Coupling and Direct Coupling, respectively. This parameter is critical since it affects the frequency content of the signals. The majority of signals are made up of both AC and DC components. The DC component is the 0 Hz component that functions as a time domain offset. For more detail, AC coupling eliminates DC offset by connecting a DC-blocking capacitor in series with the signal. AC coupling effectively rejects the signal's DC component, standardizing it to a mean of zero. On the other hand, DC (direct coupling) allows both alternating current and direct current signals to travel across a link. The DC component is a 0 Hz signal that functions as an offset for the AC component of the signal [2].

Calculation:

$$C = \frac{1}{2\pi f} = \frac{1}{2\pi * 5 * 10^5} = 318 \text{ nF}$$

Select C = 100 nF

4.1.3 Voltage Gain Amplifier

An amplifier is a type of electronic device that may amplify the amplitude of a signal (a time-varying voltage or current). It is a two-port electrical circuit that uses electricity from a power source to enhance the amplitude of a signal supplied to its input terminals, resulting in a higher amplitude signal at its output. Gain is the ratio of output voltage to input voltage [3]. The figure below is the non-inverted op amp configuration and the gain will be calculated by.

$$A_v = 1 + \frac{R_f}{R_2}$$

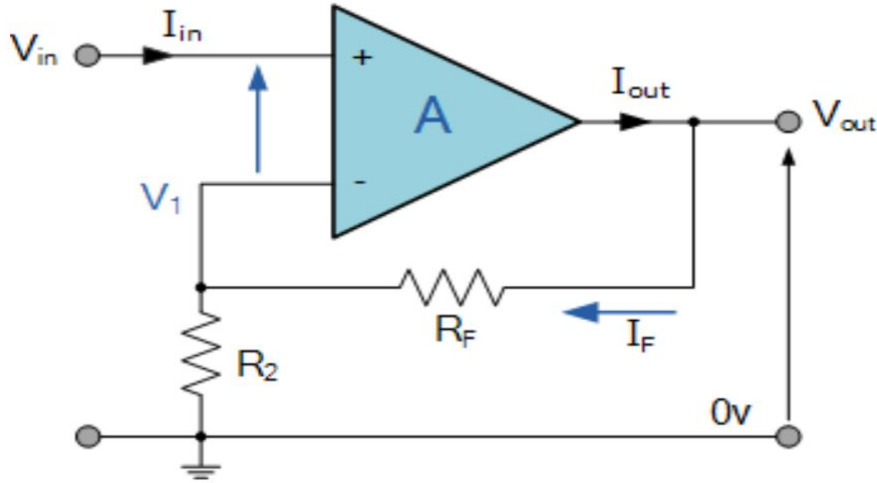


Figure 8: Non-inverted op amp configuration

In order to meet the input requirement for the ADC of Microcontroller which is working from 0 to 3.3 Voltage. The input signal will be divided into certain range as the table #1.

Setting	Range	Peak to Peak	Attenuation		Amplification	
[V/Div]	[+/-V]	[V]		[V]		[V]
0.01	0.04	0.08	1.25	0.064	10	0.64
0.02	0.08	0.16	1.25	0.128	10	1.28
0.05	0.2	0.4	1.25	0.32	10	3.2
0.1	0.4	0.8	1.25	0.64	5	3.2
0.2	0.8	1.6	1.25	1.28	2.5	3.2
0.5	2	4	1.25	3.2	1	3.2
1	4	8	12.5	0.64	5	3.2
2	8	16	12.5	1.28	2.5	3.2
5	20	40	12.5	3.2	1	3.2

Table 1: Amplification vs Voltage Range

Calculation:

- **1:1 Path**

$$A_v = 1 + \frac{R_f}{R_2} = 1 + \frac{6.2}{1000} \sim 1$$

Select: $R_f = 6.2 \Omega$

$$R_2 = 1 k\Omega$$

- **1:2.5 Path**

$$A_v = 1 + \frac{R_f}{R_2} = 1 + \frac{1500}{1000} = 2.5$$

Select: $R_f = 1.5 \text{ k}\Omega$

$$R_2 = 1 \text{ k}\Omega$$

- 1:5 Path

$$A_v = 1 + \frac{R_f}{R_2} = 1 + \frac{3900}{1000} \sim 5$$

Select: $R_f = 3.9 \text{ k}\Omega$

$$R_2 = 1 \text{ k}\Omega$$

- 1:10 Path

$$A_v = 1 + \frac{R_f}{R_2} = 1 + \frac{9100}{1000} \sim 10$$

Select: $R_f = 9.1 \text{ k}\Omega$

$$R_2 = 1 \text{ k}\Omega$$

4.1.4 DC Offset using PWM Method

DC offset is an average amplitude deviation from zero. When the signal goes out of the range ADC of the oscilloscope, the DC offset adds the Dc voltage to the incoming signal to move it in the range [1].

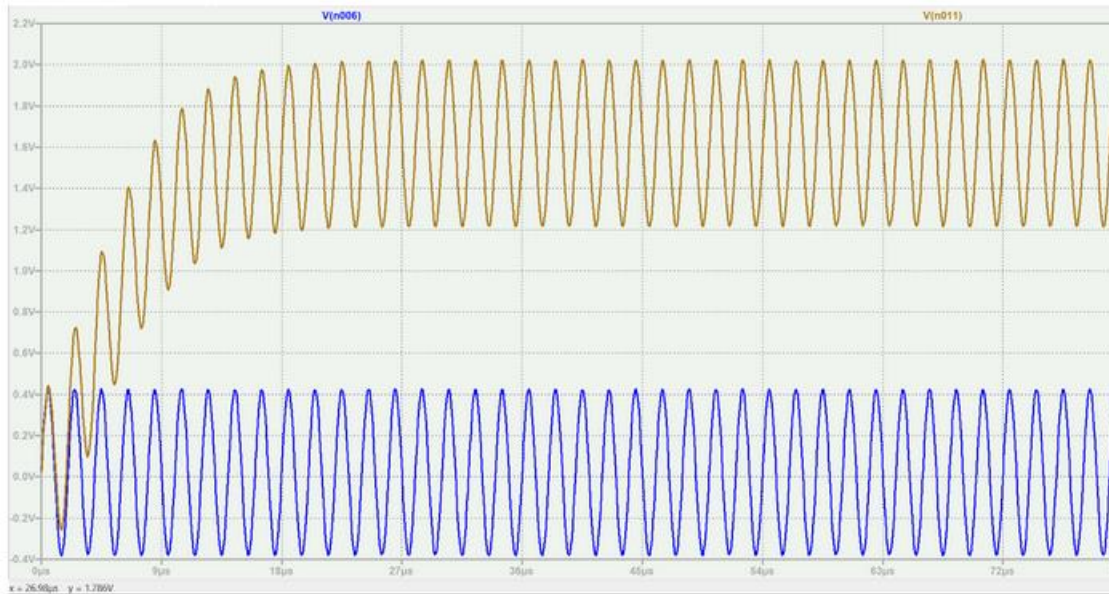


Figure 9: Range signal by analog offset

For many Microcontrollers, the default output is pulse width modulation. It may be necessary to convert a PWM to a DC signal in order to communicate with an external circuit or device because that circuit or device may need a variable DC output.

Desired DAC Voltage = $A * \text{duty cycle}$

with A is the peak value of the square wave and duty cycle = $(\text{width} / \text{period}) * 100$

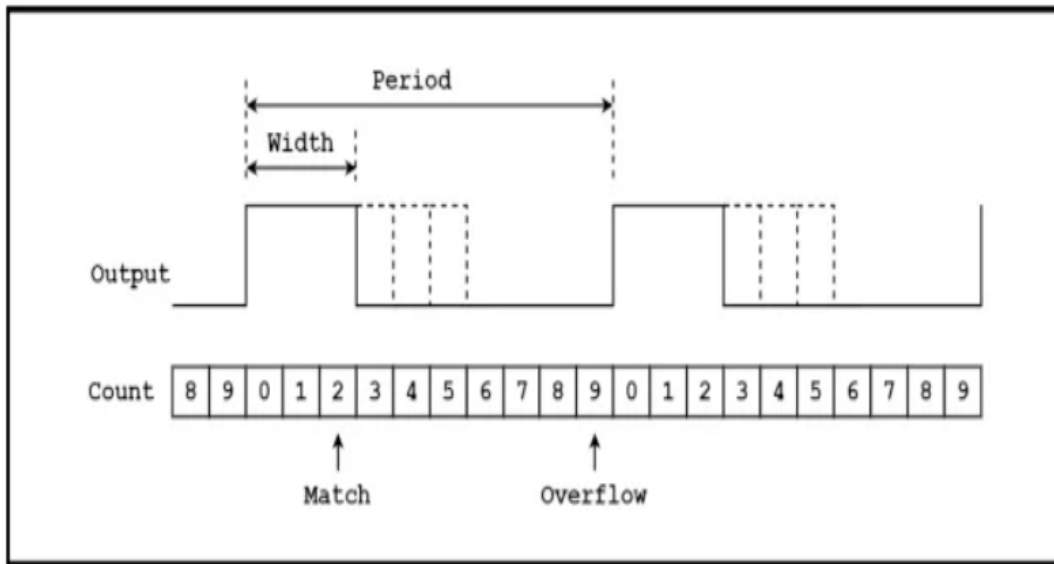


Figure 10: PWM timing diagram

PWM signal across RLC low pass Filter to invert DC signal

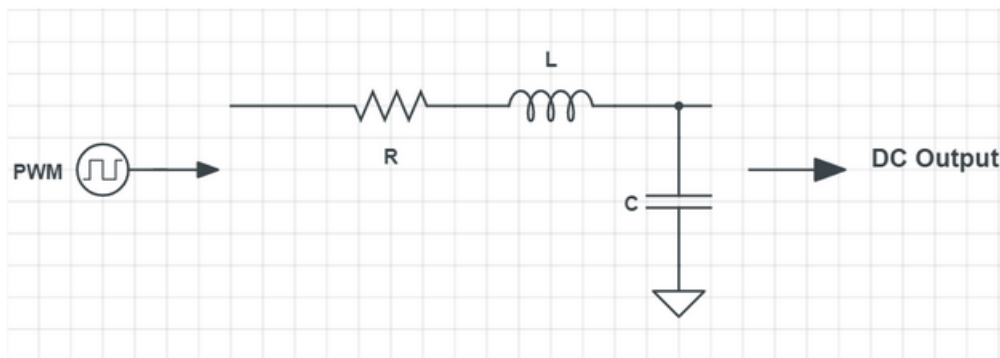


Figure 11: DC voltage by using PWM

$$f_c = \frac{1}{2\pi\sqrt{LC}}$$

$$Z = \sqrt{(X_L - X_C)^2 + R^2}$$

$$X_C = \frac{1}{2\pi fC}$$

$$X_L = 2\pi fL$$

In the case critically damped, $\zeta=1$ the following relation between R, L, C must be true:

$$R = 2\sqrt{\frac{L}{C}}$$

Choose, $L= 10\mu\text{H}$ and $C = 1\mu\text{F}$, so $R=6.2\Omega$ and $f_c= 50\text{ kHz}$.

The variable DC signal output depends on the duty cycle of the PWM signal.

4.1.5 Unity Gain Buffer

The output voltage follows or tracks the input voltage, the amplifier is a unity gain buffer. A voltage buffer amplifier's voltage gain may be unity, and it gives significant current gain.

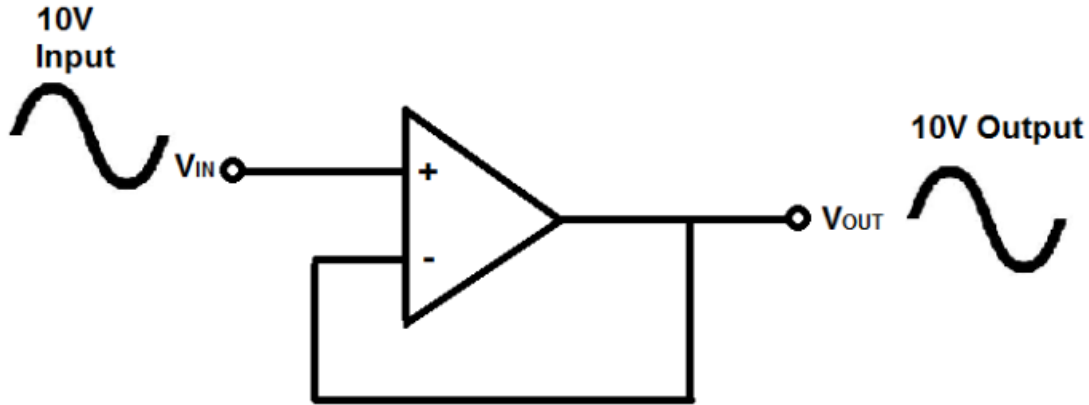


Figure 12: Unity Gain Buffer configuration

4.1.6 RC Low Pass Filter

The low pass filter (LPF) is a filter that allows the passing of the signal with low frequency and blocks the signal with high frequencies [2].

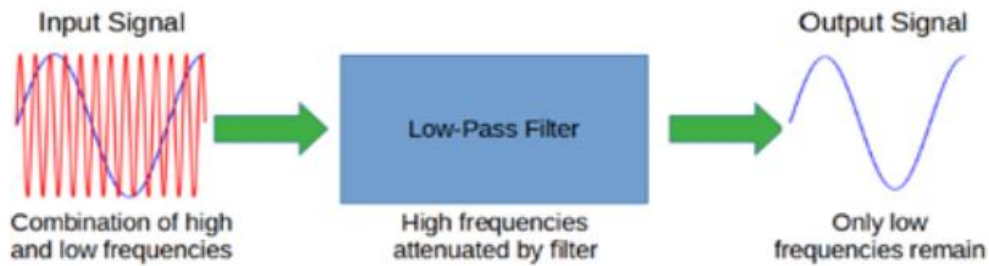


Figure 13: Low Pass Filter

$$f_c = \frac{1}{2\pi RC} \quad C = \frac{1}{2\pi R f_c} \quad R = \frac{1}{2\pi C f_c}$$

$$X_C = \frac{1}{2\pi f C} \quad z = \sqrt{X_C^2 + R^2} \quad V_{out} = \frac{A * X_C}{z}$$

$$f_c = 10\text{MHz} \quad R = 100\Omega \quad C = 160\text{pF} \text{ and } A = 5\text{V}$$

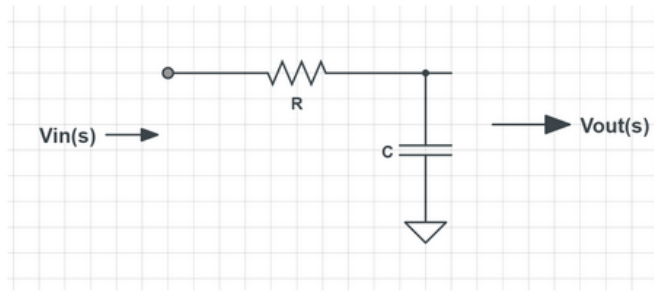


Figure 14: RC low pass filter

frequency	Xc	Z	Vout
10	99471839.43	99471839.43	5
50	19894367.89	19894367.89	5
100	9947183.943	9947183.944	5
1000	994718.3943	994718.3994	4.9999999975
4000	248679.5986	248679.6187	4.999999596
10000	99471.83943	99471.8897	4.999997473
50000	19894.36789	19894.61921	4.999936836
100000	9947.183943	9947.686585	4.999747357
500000	1989.436789	1991.948477	4.993695398

Table 2: Frequency vs V_{out} of lowpass filter simulation

Figure 15: Bode Plot diagram

4.2 MCU

- The microcontroller is like a computer integrated on a chip and used to control electronic devices. It is a self-contained embedded system with peripherals, processors, and memory.

- CPU is considered the brain of the microcontroller. It has the function of fetching and decoding instructions.
- Input/output port is used to control and communicate with devices such as monitors, LEDs, printers, etc.
- In microcontrollers memory is used to store data and programs.
- Serial port is the part that connects the microcontroller and other external peripherals.
- Timer and counter are a part of the clock speed of MCU, and it can be adjustable in configuration. It can be used as a delay or creating an event.

4.2.1 ADC Converters

In a digital oscilloscope, the analog signal is at the input, after attenuation/amplification and preliminary filtering such as for ac coupling, goes to an analog-to-digital converter (ADC). Each channel has a separate ADC, each with an external clock. Analog-to-digital conversion takes place by means of sampling. The samples are taken at specific rates and the amplitude of each sample is measured and stored. Intuitively, we know that the accuracy of the digital signal at the output depends upon the number of samples that are taken.

4.2.2 DMA

Direct Memory Access (DMA) is a capability provided by some computer bus architectures that allows data to be sent directly from an attached device (such as a disk drive) to the memory on the computer's motherboard. The microprocessor is freed from involvement with the data transfer, thus speeding up overall computer operation.

4.2.3 STM32CubeIDE

STM32CubeIDE is an all-in-one multi-OS development tool, which is part of the STM32Cube software ecosystem. STM32CubeIDE is an advanced C/C++ development platform with peripheral configuration, code generation, code compilation, and debug features for STM32 microcontrollers and microprocessors. It is based on the Eclipse/CDT framework and GCC toolchain for the development, and GDB for the debugging.

4.2.4 HAL Driver

The Hardware Abstraction Layer (HAL) driver layer provides a simple, generic multi-instance set of APIs (application programming interfaces) to interact with the upper layer (application, libraries and stacks). The HAL driver APIs are split into two categories: generic APIs, which provide common and generic functions for all the STM32 series and extension APIs, which include specific and customized functions for a given line or part number. The HAL drivers include a complete set of ready-to-use APIs that simplify the user application implementation. For example, the communication peripherals contain APIs to initialize and configure the peripheral, manage data transfers in polling mode, handle interrupts or DMA, and manage communication errors. The HAL drivers are feature-oriented instead of IP-oriented. For example, the timer APIs are split into several categories following the IP functions, such as basic timer, capture and pulse width modulation (PWM). The HAL driver layer implements run-time failure detection by checking the input values of all functions. Such dynamic checking enhances the

firmware robustness. Run-time detection is also suitable for user application development and debugging.

4.2.5 LL Driver

Low Layer (LL) drivers offer a fast light-weight expert-oriented layer which is closer to hardware than HAL. It means the LL driver offers low-level APIs at register level, such as a set of functions to initialize peripheral main features according to the parameters specified in data structures, a set of functions to reinitialize peripheral without recompiling. To use LL drivers, they require deep knowledge of the MCU and peripherals specifications.

4.3 GUI

For the GUI section, we intend to develop the GUI design based on the PyQtGraph. When we connect the Nucleo board and front end to a PC via USB, we run a python script that we develop to launch the GUI. In this script, we will collect the live data using PySerial/PyUSB and we will plot the waveform using PyQtGraph. PyQtGraph is a pure-python graphics and GUI library built on PyQt/PySide and NumPy. It is intended for use in mathematics/scientific/engineering applications. Despite being written entirely in python, the library is very fast due to its heavy leverage of NumPy for number crunching and Qt's Graphics View framework for fast display. PyQtGraph is distributed under the MIT open-source license.

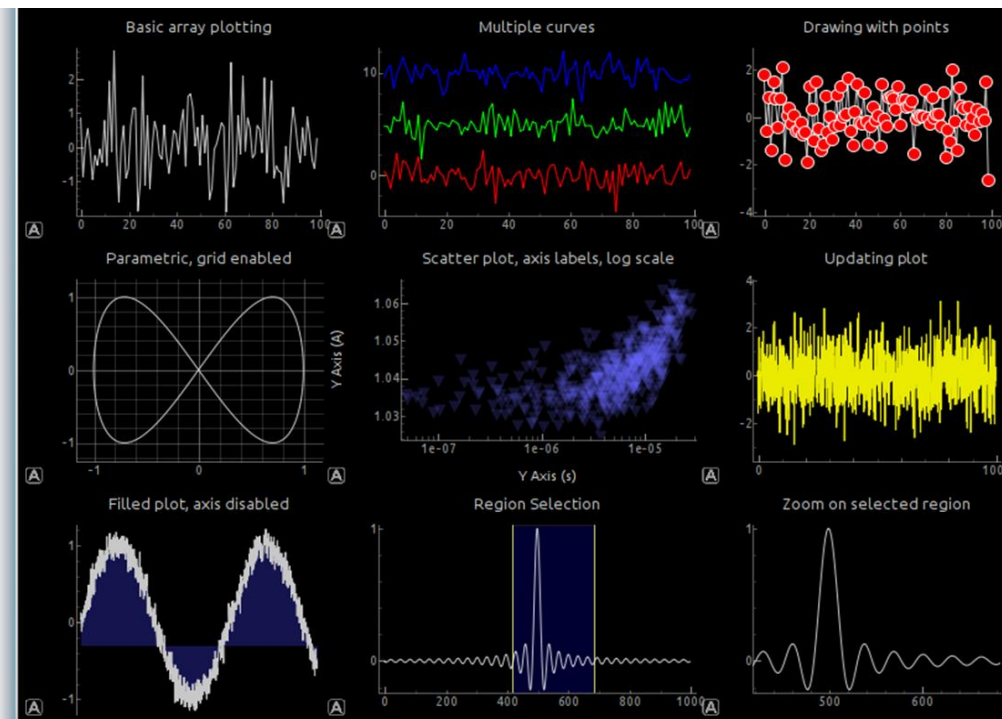


Figure 16: Main Features of PyQtGraph

PyQtGraph is known to run on Linux, Windows, and OSX. It should, however, run on any platform which supports the following packages:

- Python 3+
- PyQt 5, PyQt6, PySide2, or PySide6
- NumPy

- SciPy is optional for some numerical procedures
- python-opengl bindings are required for 3D graphics

PyQtGraph can be installed using these methods:

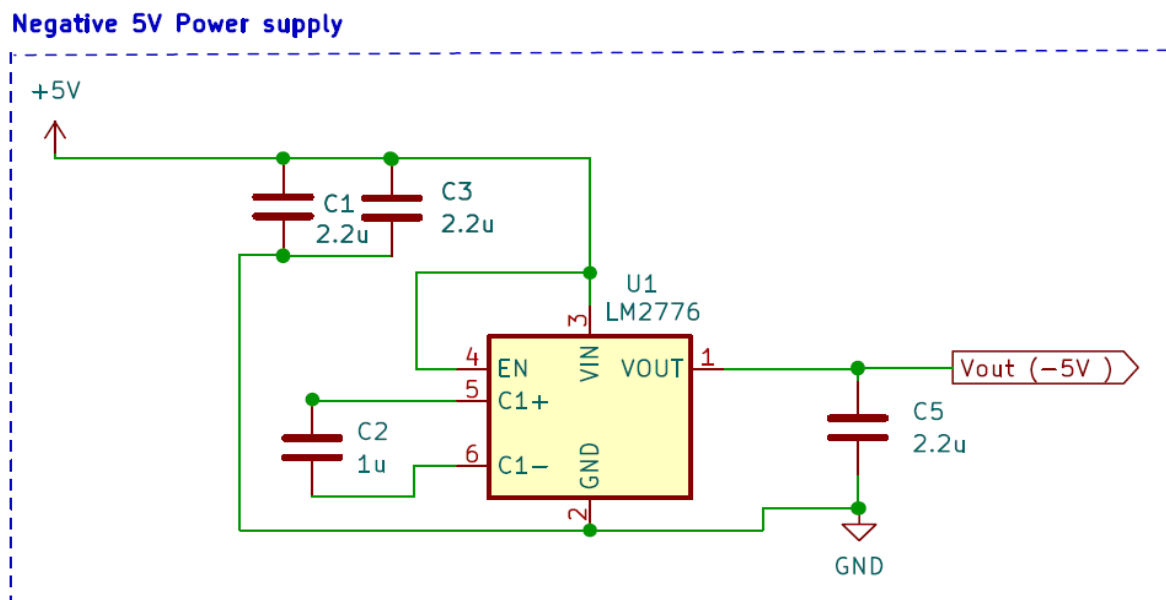
- From PyPI
 - Last released version: `pip install pyqtgraph`
- From conda
 - Last released version: `conda install -c conda-forge pyqtgraph`
- To install system-wide from source distribution: `python setup.py install`
- Many linux package repositories have released versions.
- To use with a specific project, simply copy the PyQtGraph subdirectory anywhere that is importable from your project.

We can learn PyQtGraph is to browsing through the examples. We can use this command `python -m pyqtgraph.examples` to launch the example application.

5. Detailed Design

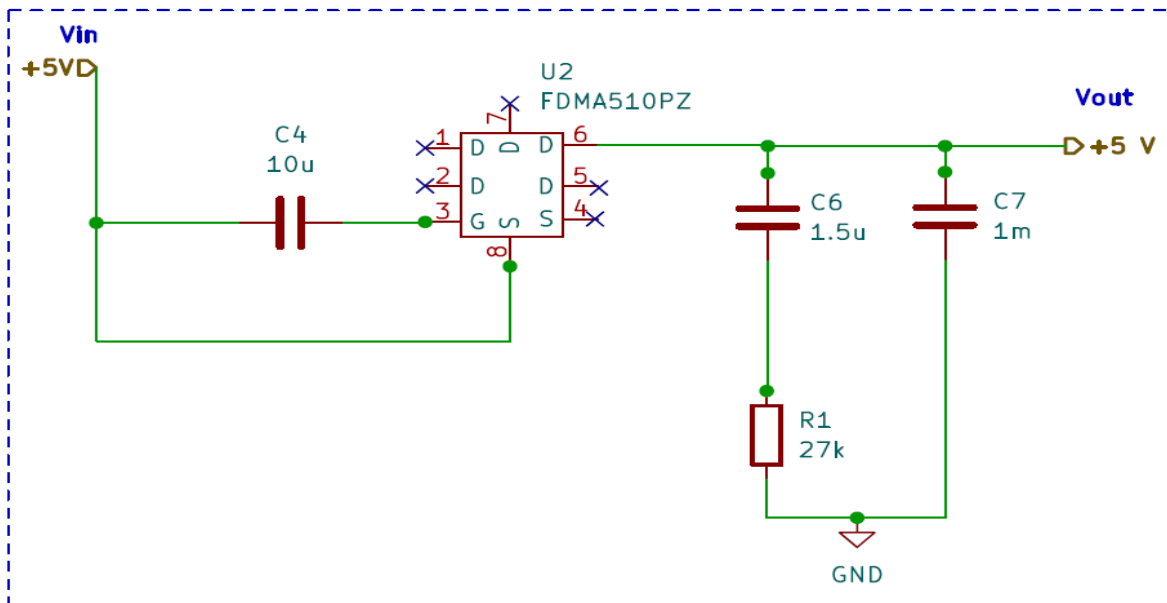
5.1 Analog Front-End Schematics

5.1.1 Power Supply -5V Schematic



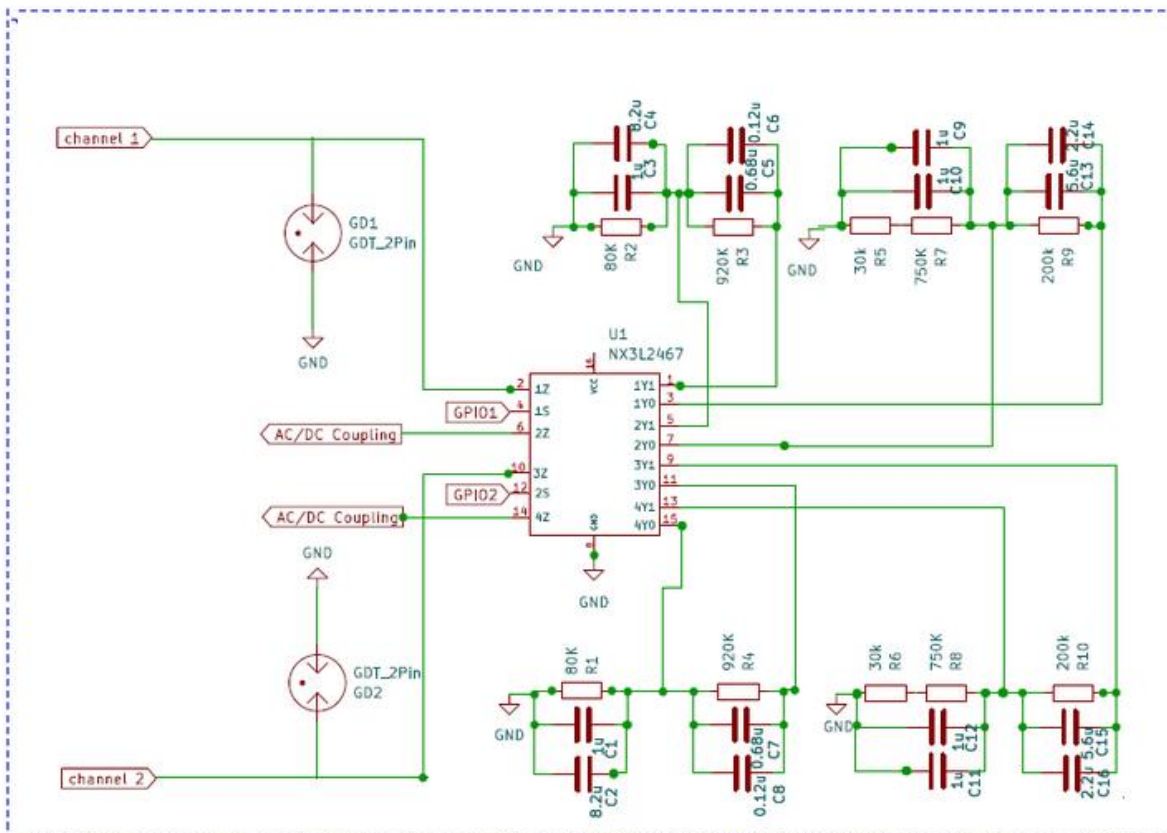
5.1.2 USB Soft Start Schematic 5.1.3 Attenuator Schematic 5.1.3 Attenuator Schematic

USB Soft Start Schematic



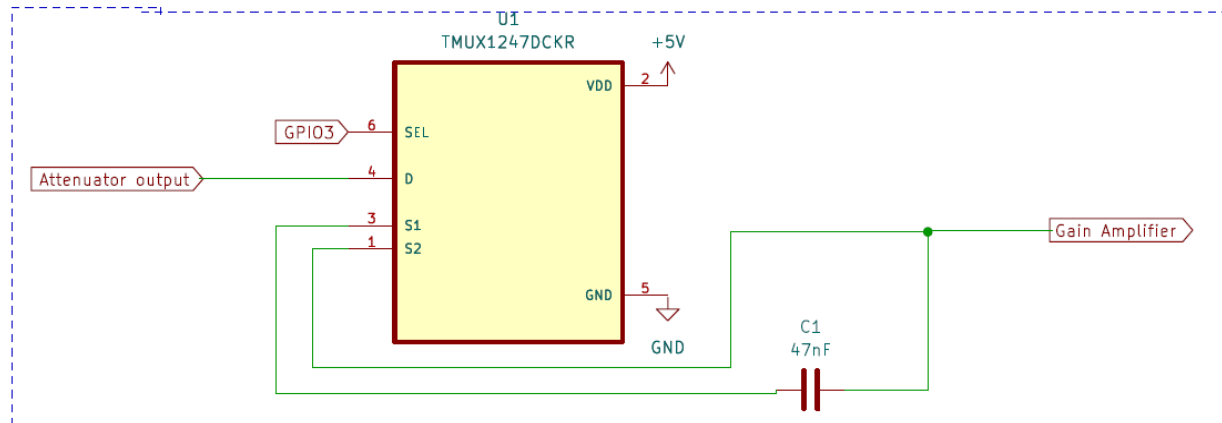
5.1.3 Attenuator Schematic

Attenuator Schematic



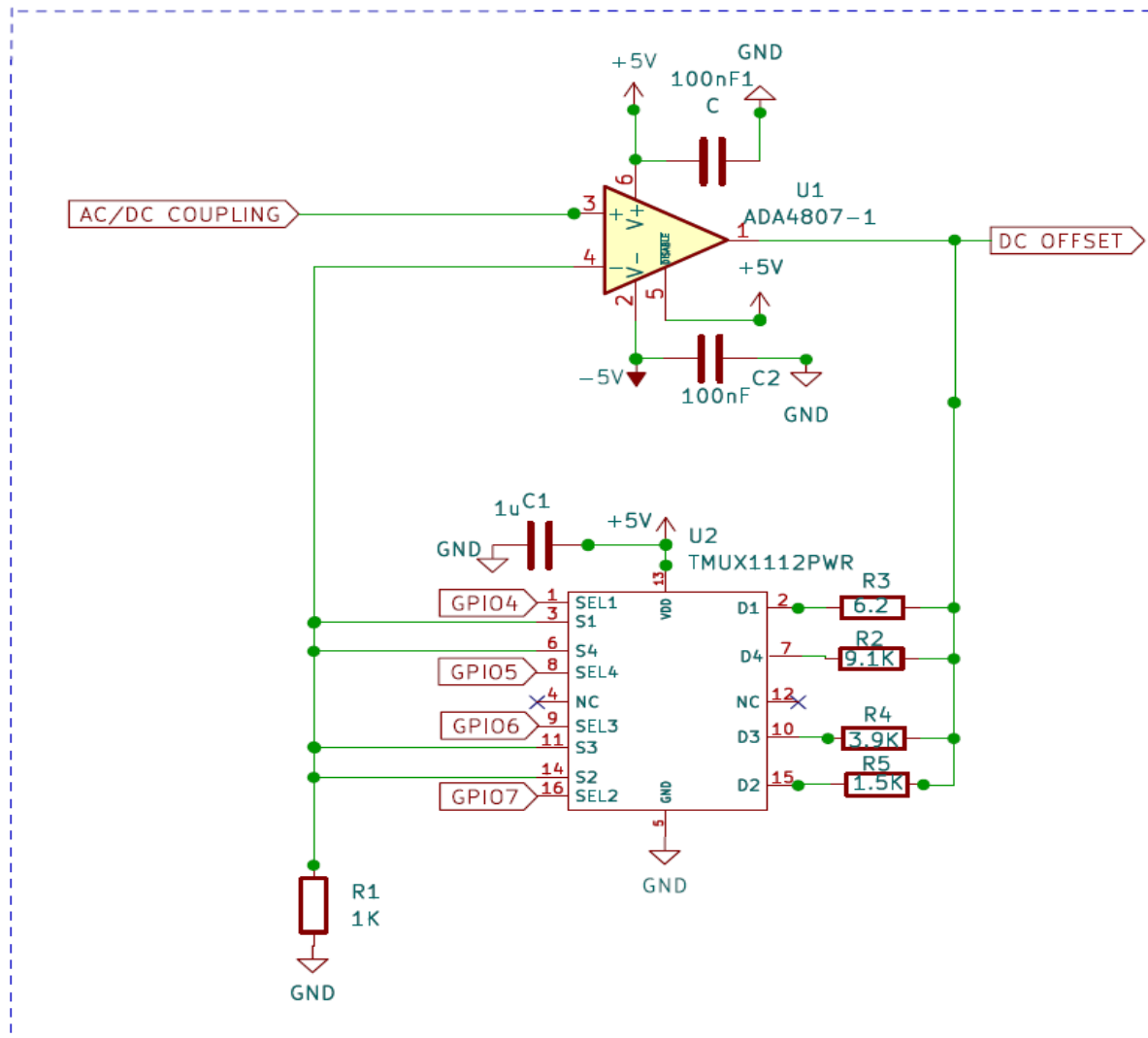
5.1.4 AC/ DC Coupling Schematic

AC/DC COUPLING



5.1.5 Gain Amplifier Schematic

GAIN AMPLIFIER

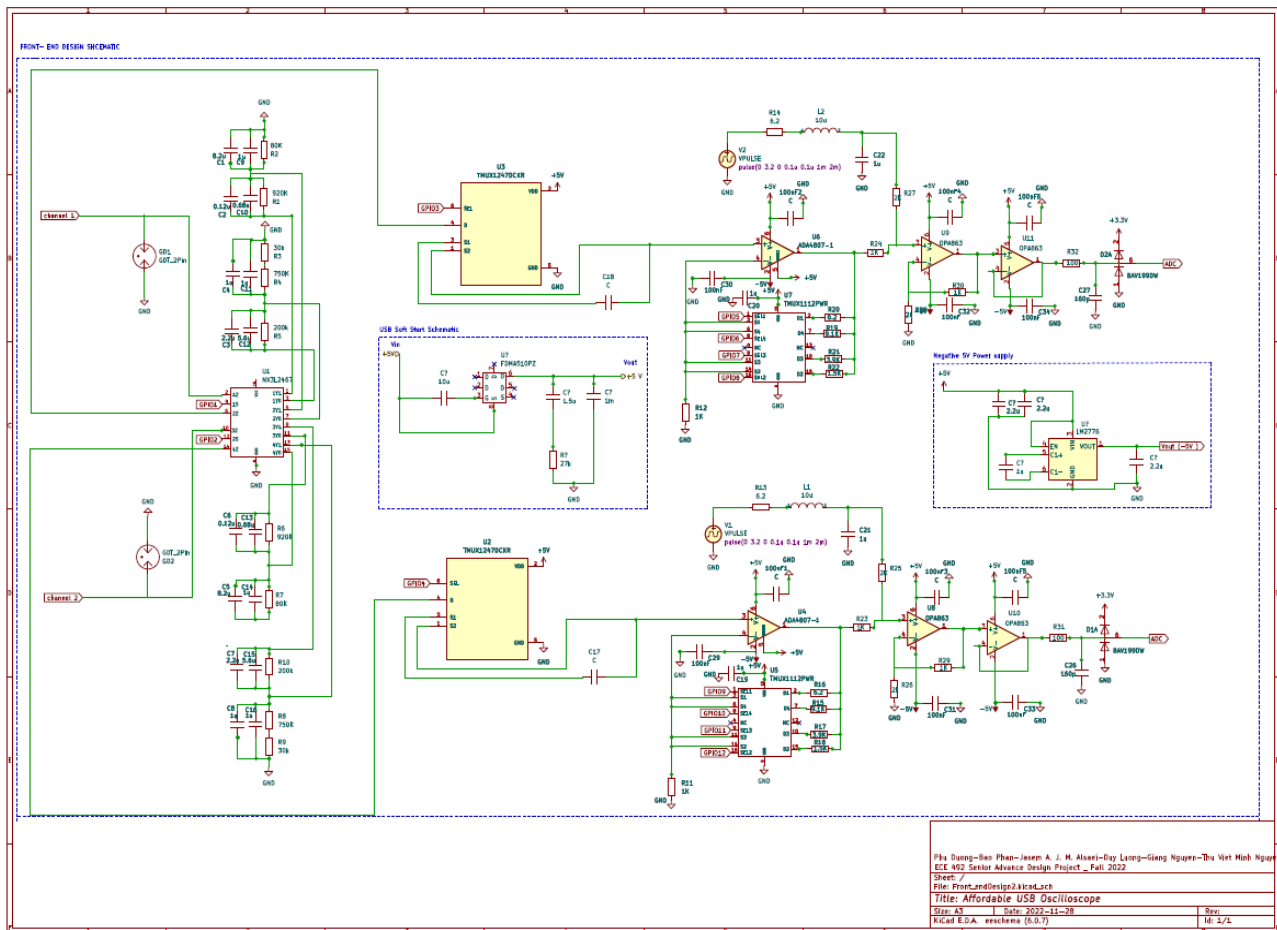


5.1.6 DC Offset using PWM, Buffer, and RC Low Pass Filter Schematic

DC OFFSET - BUFFER - LPF - VOLTAGE PROTECTION



5.1.7 The Whole Design Schematic



5.2 Analog Front-End Component Selection

5.2.1 Power Supply -5V

Components	Description	Part number	Quantity (EA)	Datasheet links
Switching Voltage Regulators	<ul style="list-style-type: none"> • Input Voltage: 2.7 V to 5.5 V • 200-mA Output Current • Inverts Input Supply Voltage • Low-Current PFM Mode Operation • 2-MHz Switching Frequency • Greater than 90% Efficiency • Current Limit and Thermal Protection 	LM2776DBVT	1	LM2776
Capacitor	1u		1	
Capacitor	2.2u		2	

5.2.2 Attenuator

Scale	Components	Description	Part number	Quantity (EA)	Datasheet links
1.25:1	Analog Switch	Analog Switch ICs DUAL LO-OHMIC DBL-PL DBL-THROW ANALOG SW	NX3L2467 PW,118	1	NX3L2467
	Capacitor	1u		2	
	Capacitor	2.2u		1	
	Capacitor	5.6u		1	
	Resistor	750k		1	
	Resistor	30k		1	
	Resistor	200k		1	
12.5:1	Capacitor	0.68u		1	
	Capacitor	0.12u		1	
	Capacitor	8.2u		1	
	Capacitor	1u		1	

	Resistor	910k		1	
	Resistor	10k		1	
	Resistor	75		1	
	Resistor	5.1k		1	

5.2.3 AC/ DC Coupling

Components	Description	Part number	Quantity (EA)	Datasheet links
Analog Switch	TMUX1247 5-V Bidirectional, 2:1 (SPDT) General Purpose Switch	TMUX1247DCKR	1	TMUX1247
capacitor	47n		1	

5.2.4 Gain Amplifier

Components	Description	Part number	Quantity (EA)	Datasheet links
Switches	TMUX111x 5-V, Low-Leakage-Current, 1:1 (SPST), 4-Channel Precision Switches	TMUX111 2PWR	1	TMUX1112
Operational Amplifiers	Operational Amplifiers - Op Amps 180MHz, 3.1 nV/Hz, 1 mA, RRIO Opamp	ADA4807-1ARJZ-R7	1	ADA 4807-1
Resistor	1k		1	
Resistor	6.2		1	
Resistor	3.9k		1	
Resistor	9.1k		1	
Resistor	1.5k		1	
Capacitor	1u		1	

5.2.5 DC Offset using PWM

Components	Description	Part number	Quantity (EA)	Datasheet links
Operational Amplifiers	High Speed Operational Amplifiers Single-channel, low-power, 110-MHz, 12-V, RRIO voltage feedback amplifier	OPA863SI DBVR	1	OPA863
Capacitor	1u		1	
Resistor	6.2		2	
Resistor	1k		2	
Resistor	2k		2	
Inductor	10u		1	

5.2.6 Buffer and RC Low Pass Filter

Components	Description	Part number	Quantity (EA)	Datasheet links
Operational Amplifiers	High Speed Operational Amplifiers Single-channel, low-power, 110-MHz, 12-V, RRIO voltage feedback amplifier	OPA863SI DBVR	1	OPA863
Capacitor	160p		1	
Resistor	100		1	

5.3 MCU

For the detailed design of MCU, there are three parts. First, the controller layer is the microcontroller processor configuration. The function of the MCU is required for the GUI. This function will include direct memory access (DMA), ADC converter and speed, Timer, GPIO and the USB transmit/receive. STM32CubeMX and Keil will be used to support these functions. STM32CubeMX is used for setup MCU configuration, and Keil is used for coding part of the MCU. For the MCU hardware, this will include the Hardware abstraction layer (HAL) driver and the low layer (LL) driver. HAL driver is used to perform functions such as GPIO, Timer, ADC, DMA, and USB. LL driver is used for adjusting or changing in register level while the MCU is running without reinitialization by STM32CubeMX.

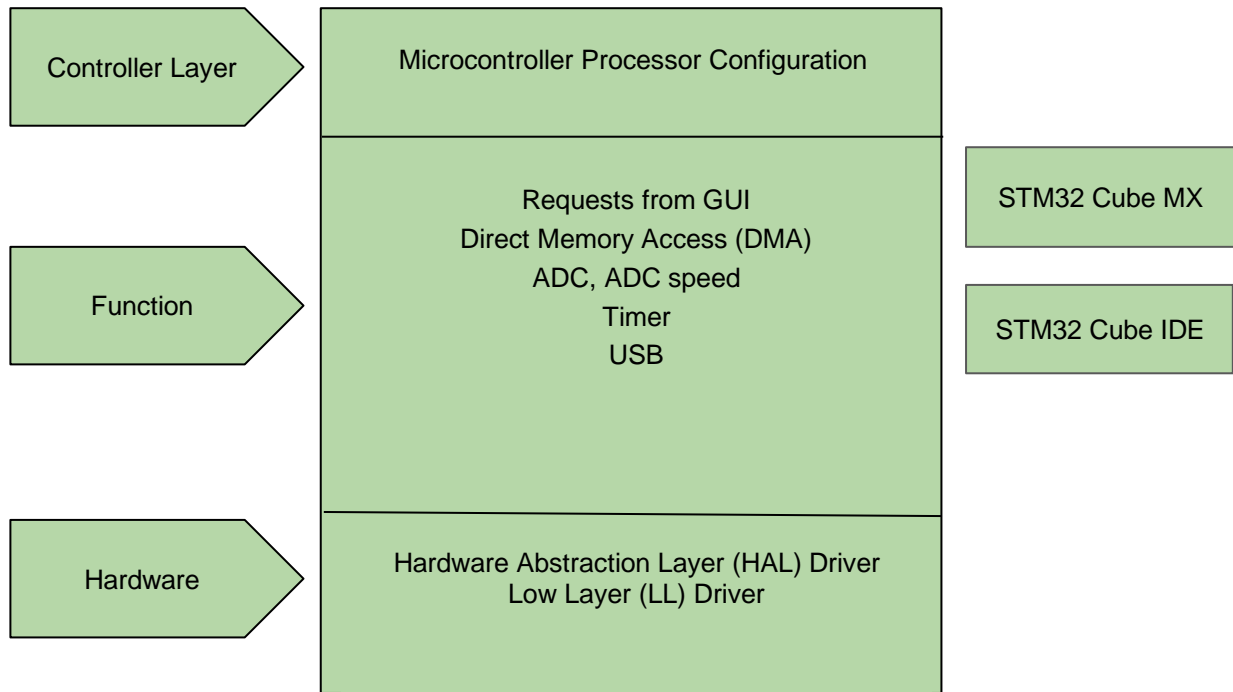


Figure 17: MCU Detailed Design

5.4 GUI Design

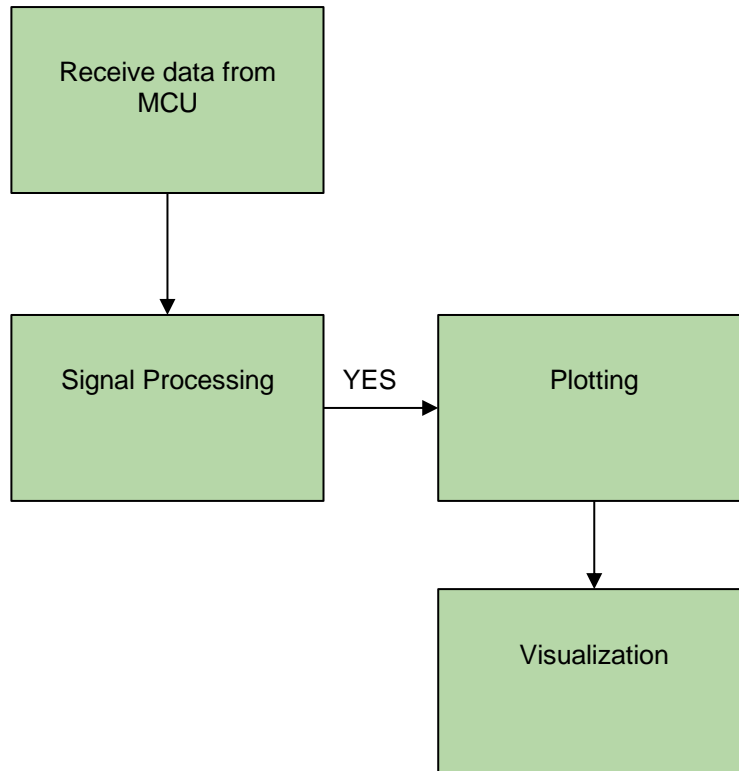


Figure 18: GUI Detailed Design

The goal for the GUI is to be able to process all the signals from the ADC to the original signal, create an array of arrays from that data, plot all that information fast enough to not notice any delay, and

finally to be able to do all that with two-channels and a functional looking GUI. The GUI will be made fully with Python including some important math and GUI libraries, and these libraries are as follows:

- NumPy/SciPy (arrays and matrix math)
- PyQt/PySide (GUI toolkit)
- PyQtGraph (fast plotting of data)

These are the main libraries that are used to make the user interface of the system. After the signal is received from the ADC, the computer will use Python3+ with NumPy/SciPy to rearrange the data into an array or arrays to the same format that PyQtGraph accepts. The GUI will be made by the PyQt/PySide libraries for an easy and simple creation while still adding PyQtGraph as the plotting window. PyQtGraph has multiple plotting configurations such as a cursor function, and a zoom in function; these functions will be used to create a robust and professional looking plot with intuitive controls and all the needed information upfront. The cursor function will be used to measure the voltages at certain points. The zoom in function will be used to section off the signal and be able to move left or right similar to a real oscilloscope.

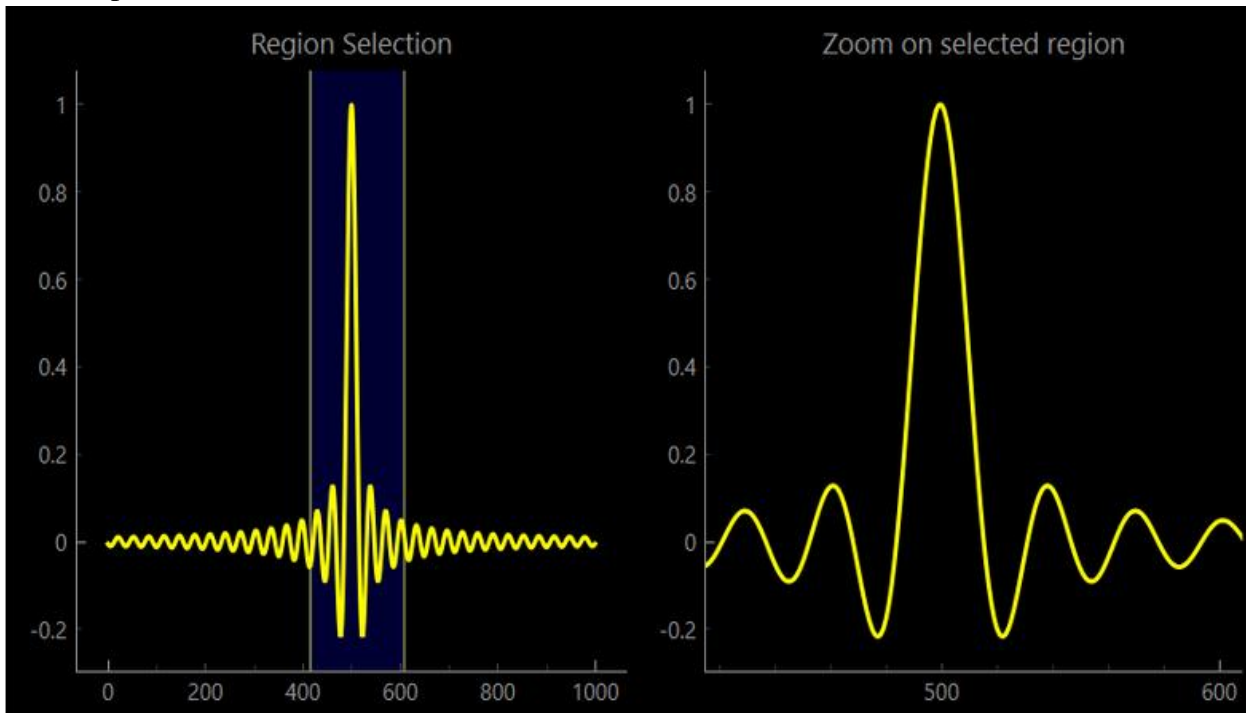


Figure 19: The zoom function in PyQtGraph

Finally, the cursor function will be added to both signals which will make it easier to know the voltages at certain points. The cursor function will be added by a PyQtGraph function to make it as easy and as fast as possible.

6. Prototyping & Early Testing Progress Report

6.1 Analog Front-End Testing

6.1.1 Attenuator Path (1.25:1) Simulation

- Gain 1:1

Testing the input with $V_{in}=1V$

The theoretical result after attenuator: $V_{out}=1/1.25=0.8V$

The theoretical result after gain amplifier: $V_{out}=0.8*1=0.8V$

The experiment $V_{out}=0.794V$

Testing on LTspice:

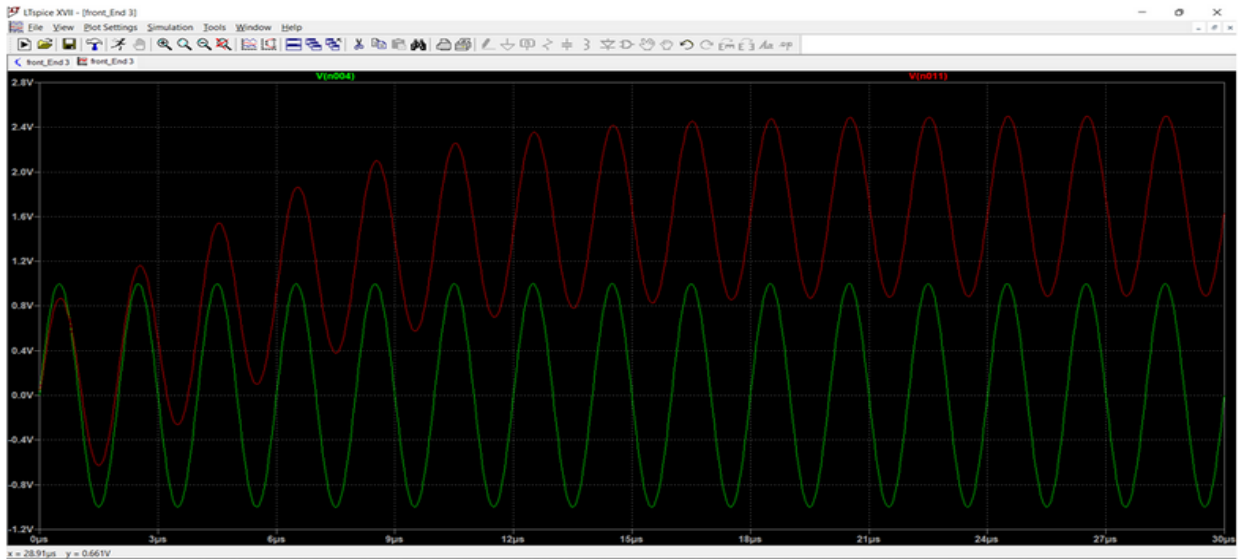


Figure 20: 1.25:1 Attenuator Path Simulation

Testing on breadboard using AD2

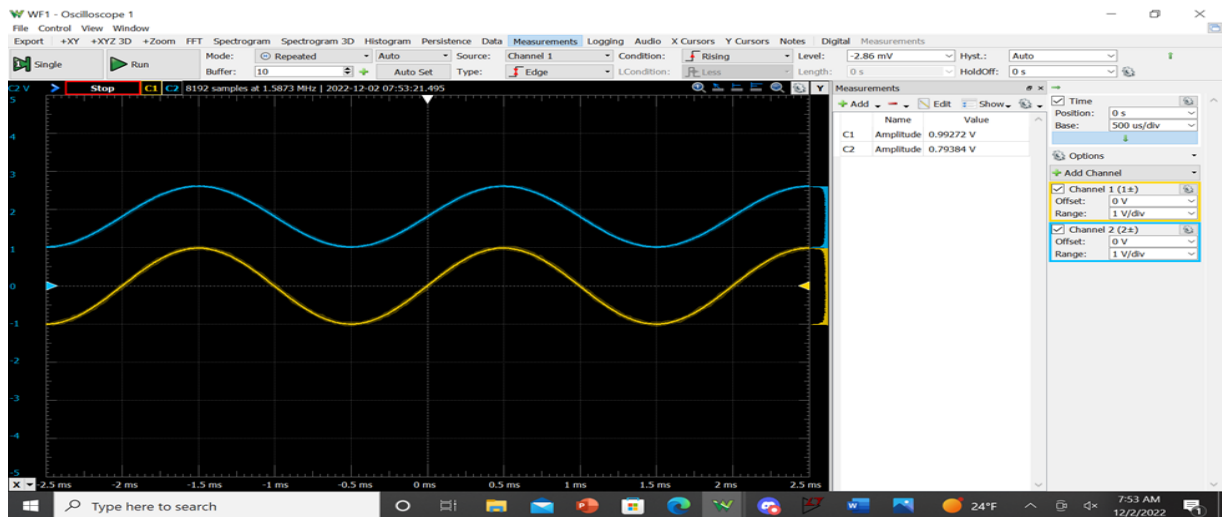


Figure 21: V_{out} vs V_{in} (AD2)

- Gain 1:2.5

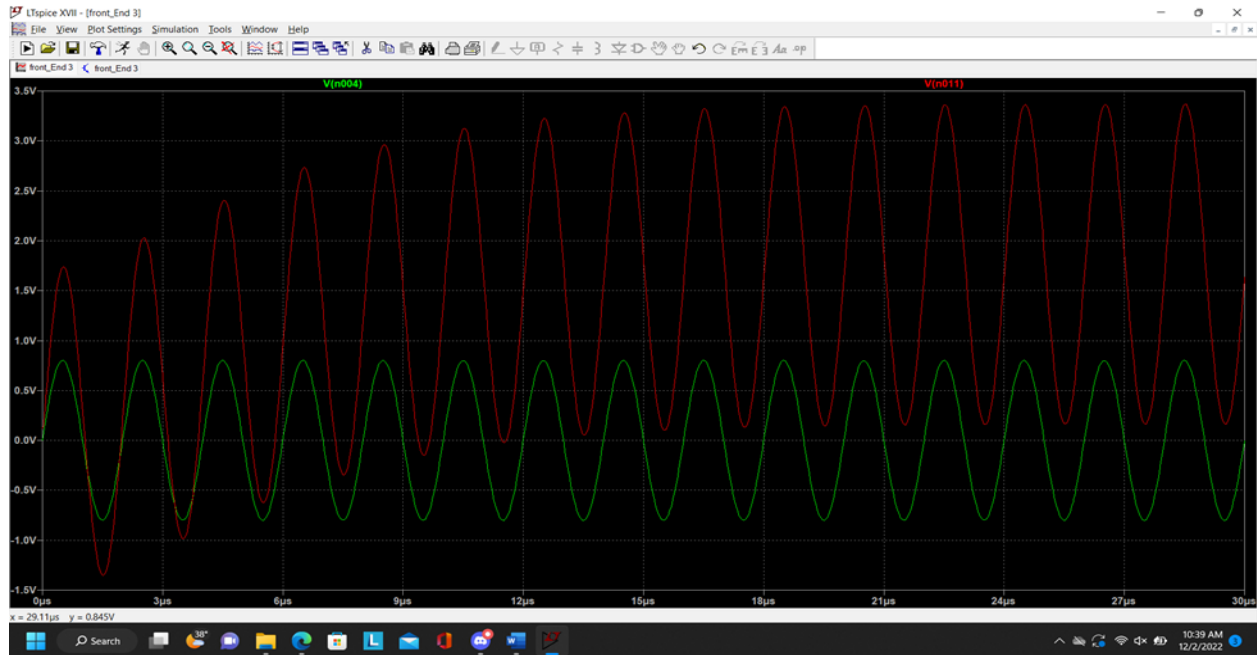
Testing the input with $V_{in}=0.78V$

The theoretical result after attenuator: $V_{out}=0.78/1.25 = 0.624V$

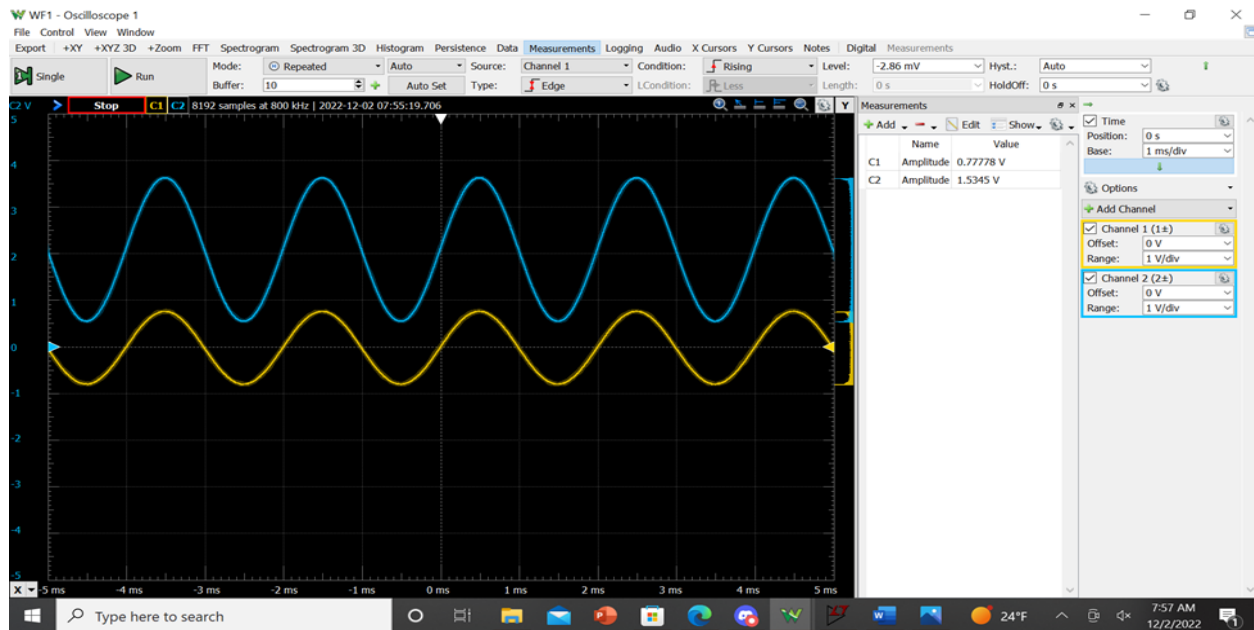
The theoretical result after gain amplifier: $V_{out} = 0.624*2.5=1.56 V$

The experiment: $V_{out} = 1.5345V$

Testing on LTspice:

Figure 22: V_{out} vs V_{in} (LT Spice)

Testing on breadboard using AD2

Figure 23: V_{out} vs V_{in} (AD2)

- Gain 1:5

Testing the input with $V_{in} = 0.378V$

The theoretical result after attenuator: $V_{out} = 0.378/1.25 = 0.3024 V$

The theoretical result after gain amplifier: $V_{out} = 0.3024 * 5 = 1.52 V$

The experiment: 1.47 V

Testing on LTspice

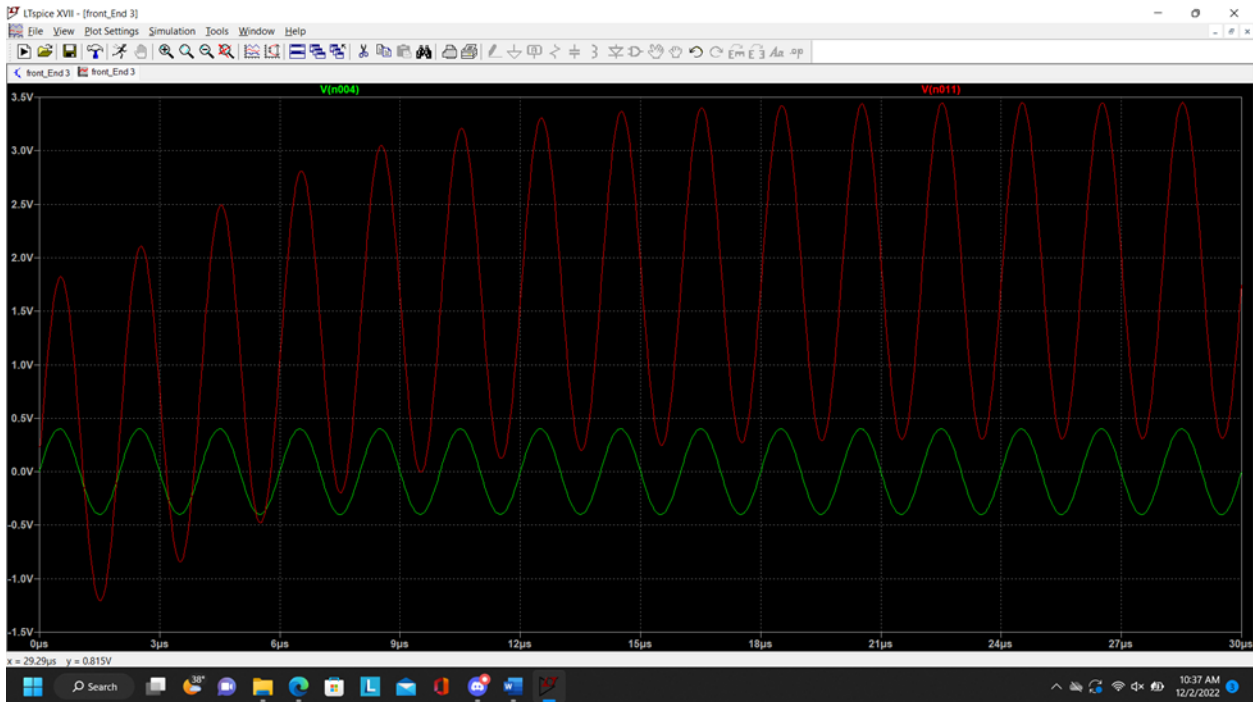


Figure 24: V_{out} vs V_{in} (LT Spice)

Testing on breadboard using AD2 using AD2

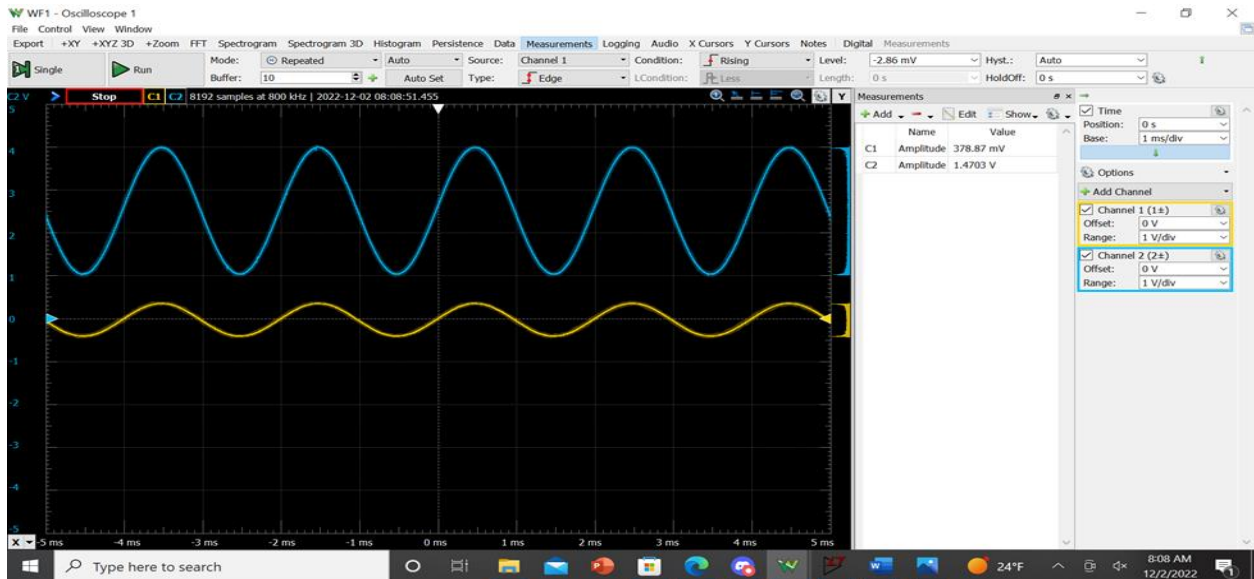


Figure 25: V_{out} vs V_{in} (AD2)

- Gain 1:10

Testing the input with $V_{in} = 0.196V$

The theoretical result after attenuator: $V_{out} = 0.196/1.25 = 0.1568V$

The theoretical result after gain amplifier: $V_{out} = 0.1568 \times 10 = 1.568V$

The experiment: 1.49 V

Testing on LTSpice:

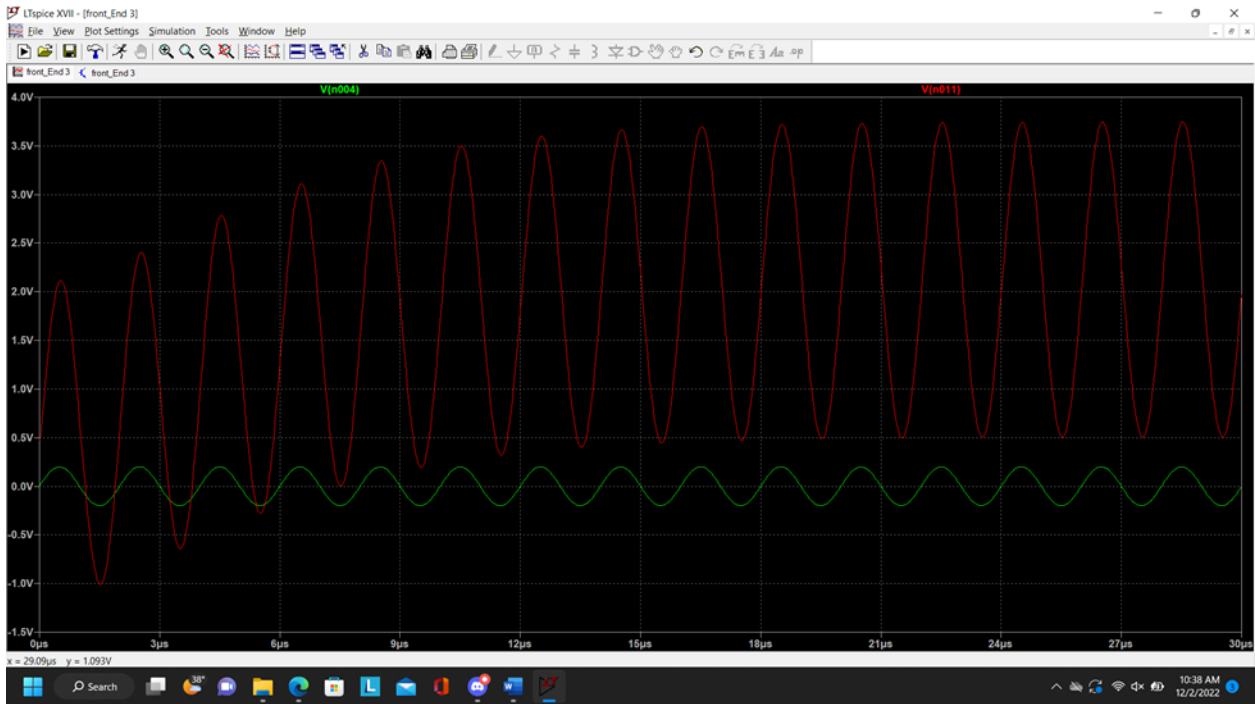


Figure 26: V_{out} vs V_{in} (LT Spice)

Testing on breadboard using AD2:

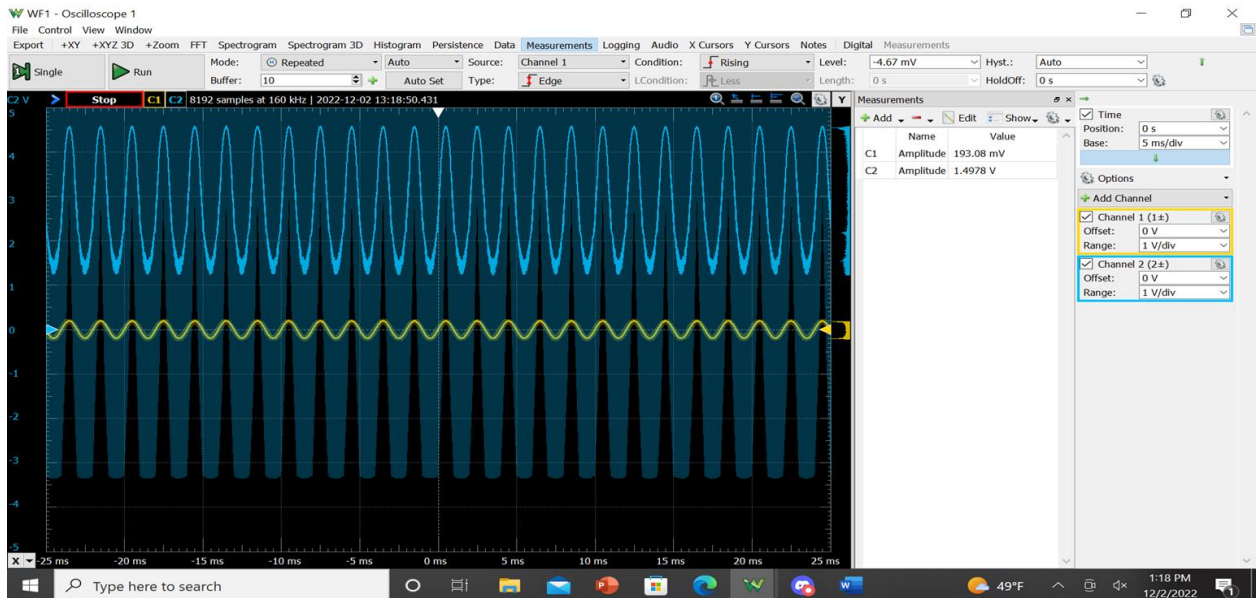


Figure 27: V_{out} vs V_{in} (AD2)

- Bandwidth

Testing on LTspice

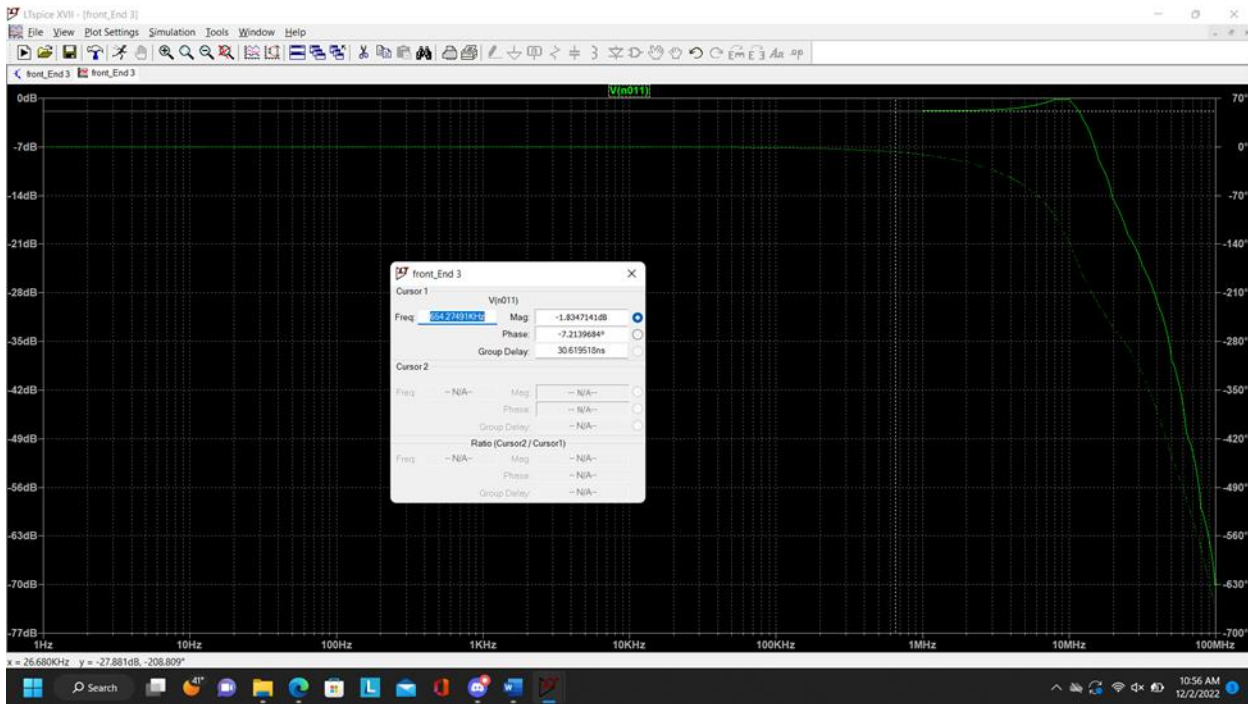


Figure 28: Bandwidth Test (LTspice)

Testing on breadboard using AD2:

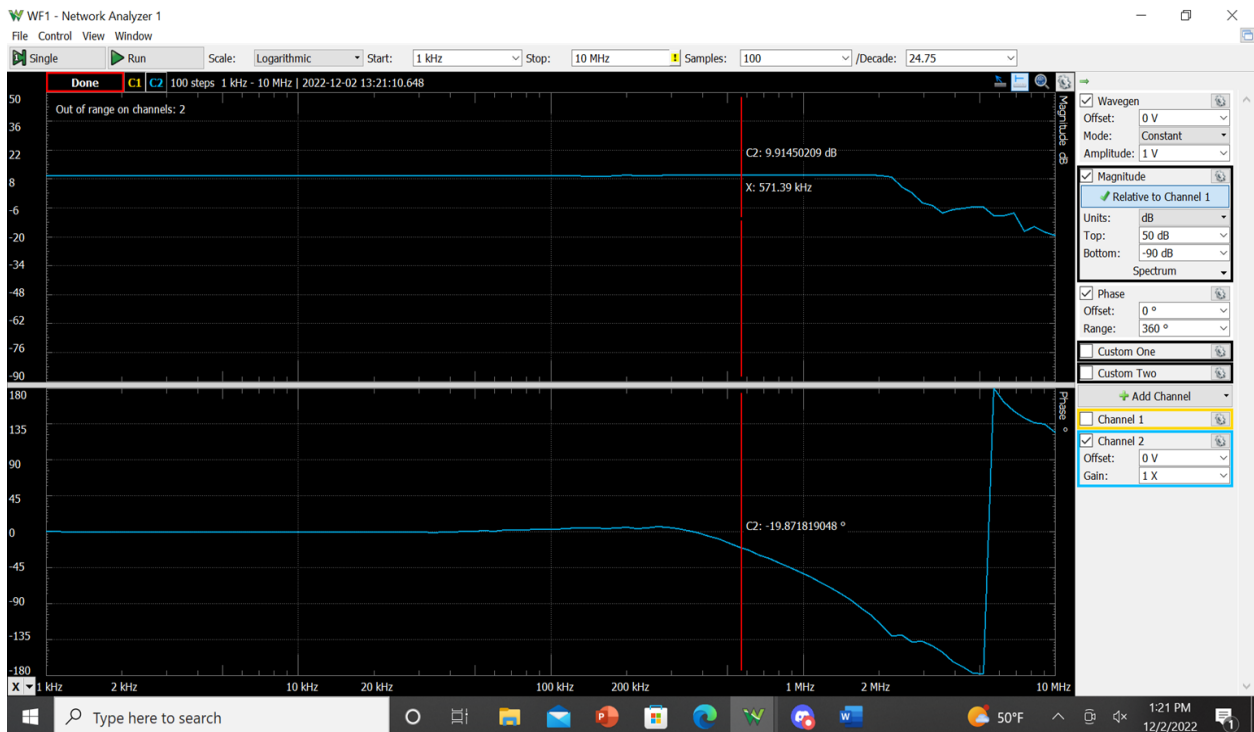


Figure 29: Bandwidth Test (AD2)

6.1.2 Attenuator Path (12.5:1) Simulation

- Gain 1:10

Testing the input with $V_{in} = 2V$

The theoretical result after attenuator: $V_{out} = 2/12.2 = 0.164V$

The theoretical result after gain amplifier: $V_{out} = 0.164 \times 10 = 1.64V$

The experiment $V_{out} = 1.6V$

Test on LTspice:

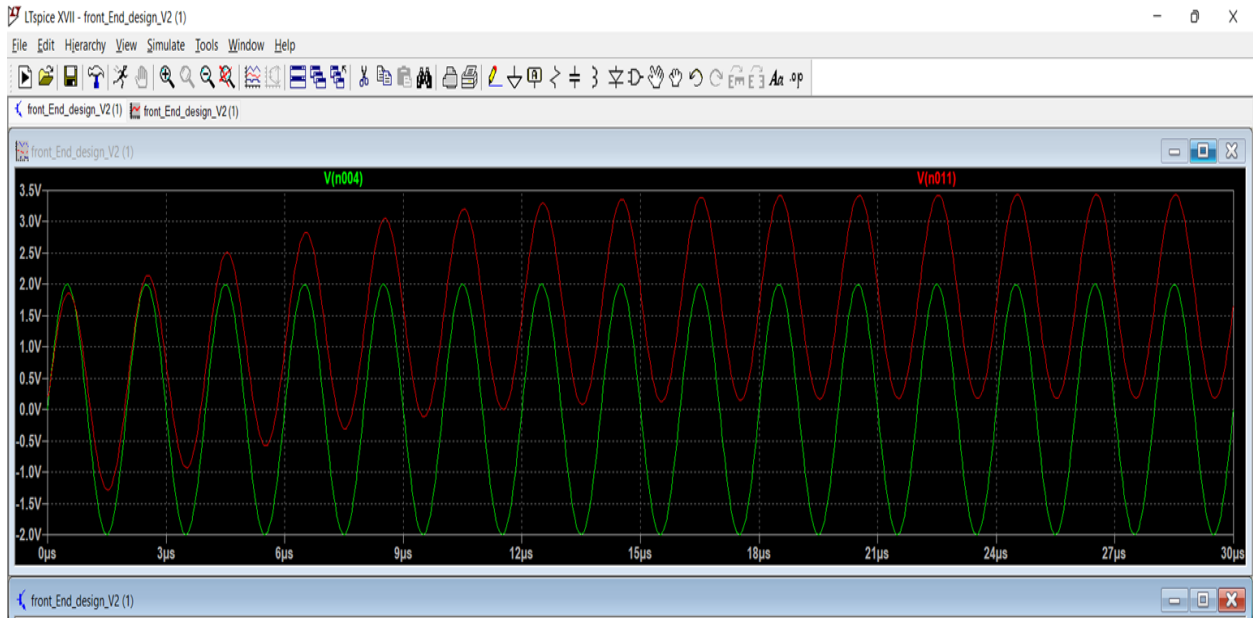


Figure 30: V_{out} vs V_{in} (LT Spice)

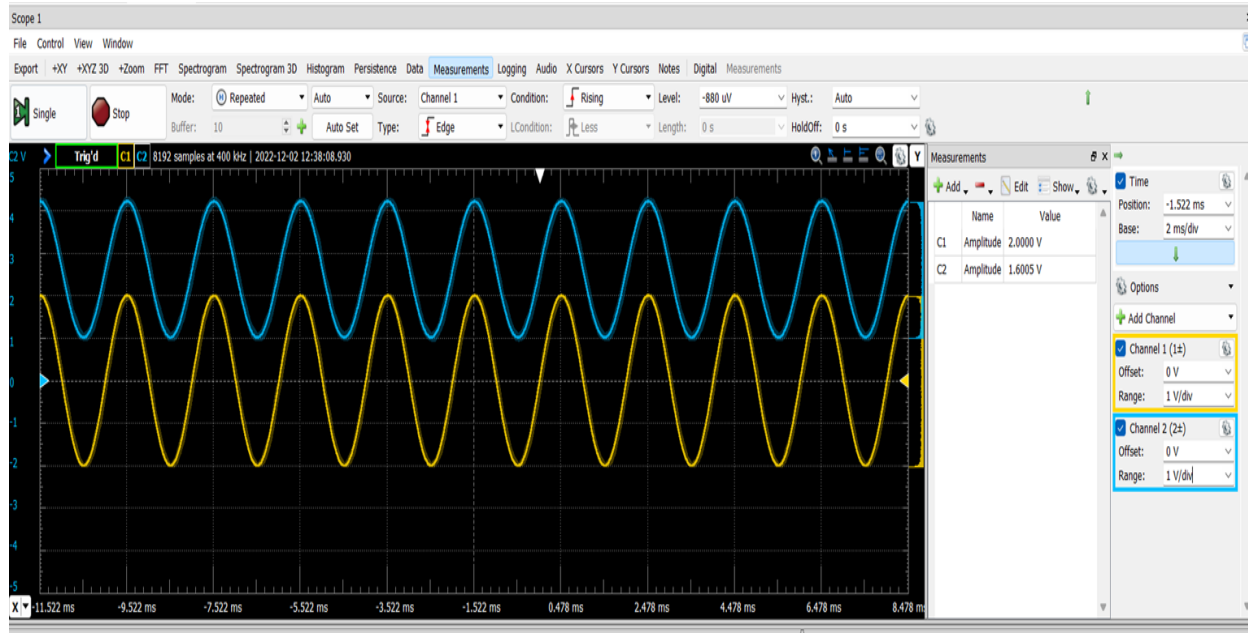


Figure 31: V_{out} vs V_{in} (AD2)

- Gain 1:5

Testing the input with $V_{in} = 4V$

The theoretical result after attenuator: $V_{out} = 4/12.2 = 0.328V$

The theoretical result after gain amplifier: $V_{out} = 0.328 * 5 = 1.639V$

The experiment $V_{out} = 1.5547V$

Test on LTspice

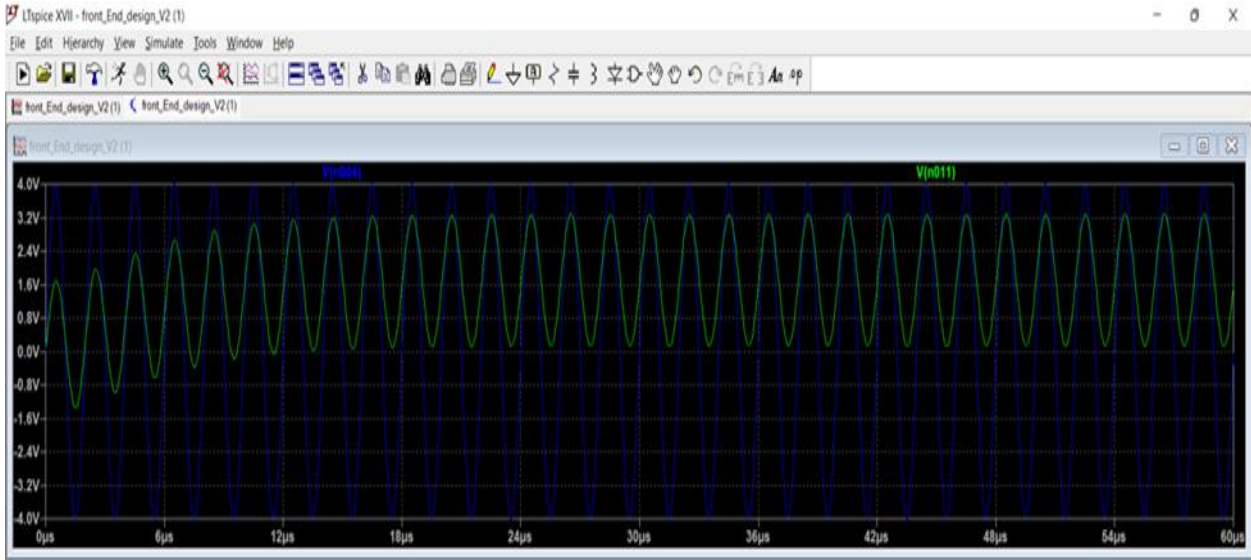


Figure 32: V_{out} vs V_{in} (LT Spice)

Test on breadboard using AD2

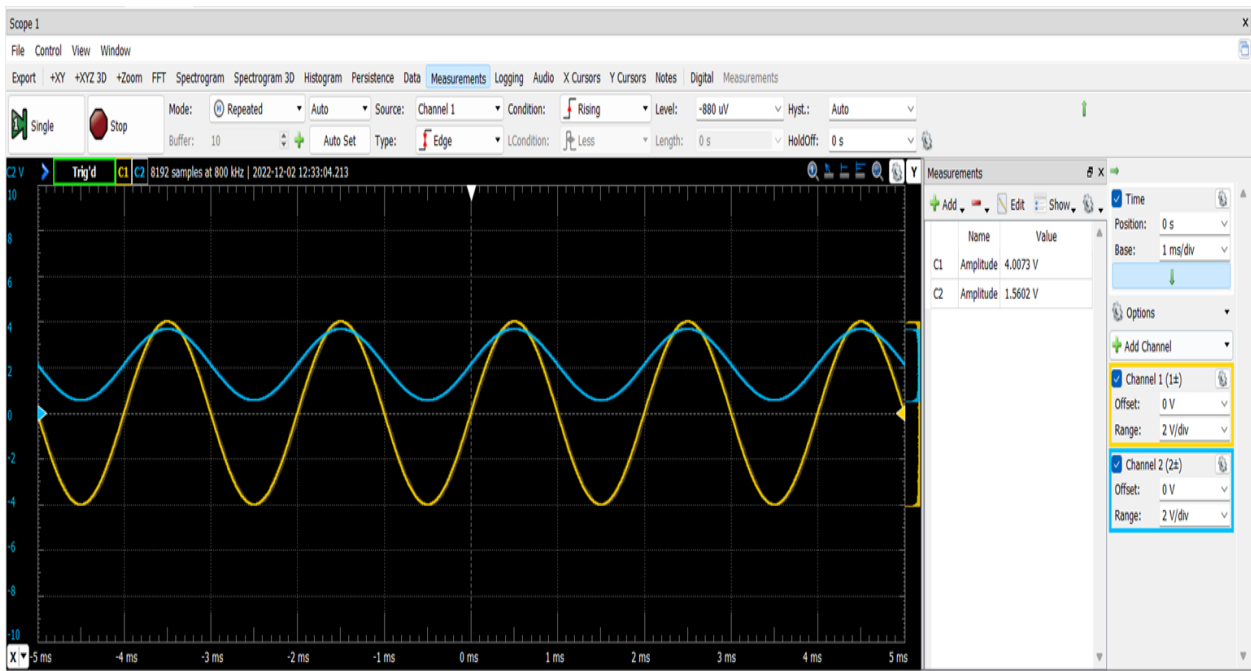


Figure 33: V_{out} vs V_{in} (AD2)

- Gain 1:2.5

Testing the input with $V_{in} = 5V$

The theoretical result after attenuator: $V_{out} = 5/12.2 = 0.4V$

The theoretical result after gain amplifier: $V_{out} = 0.4 * 2.5 = 1V$

The experiment $V_{out} = 1V$

Testing on LTspice

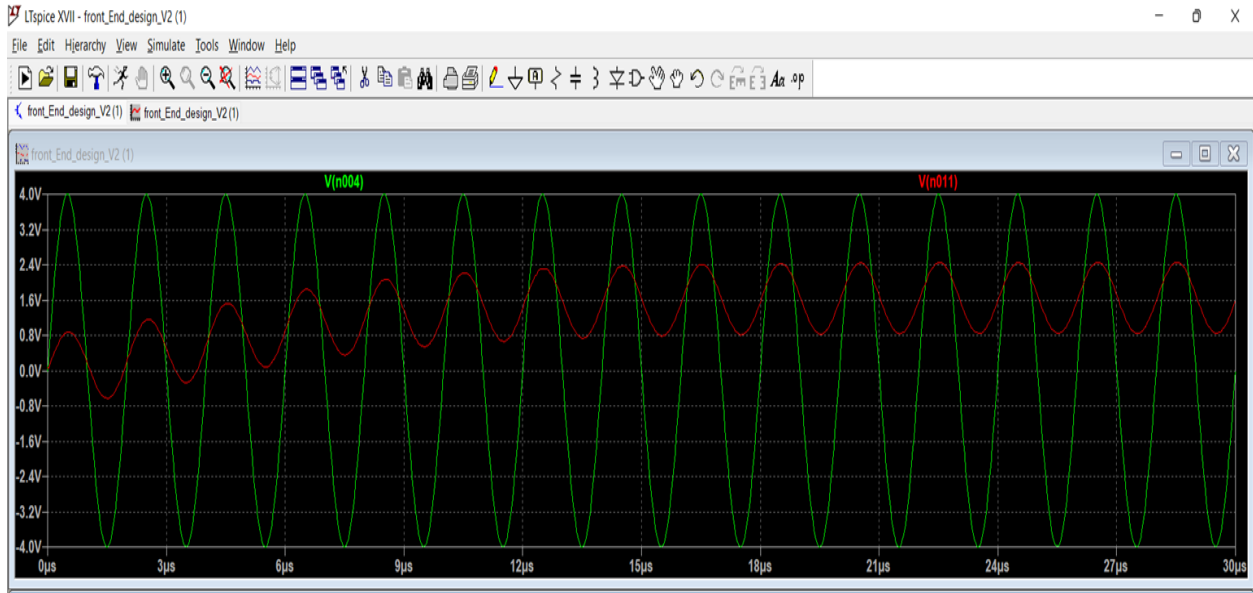


Figure 34: V_{out} vs V_{in} (LT Spice)

Testing on breadboard using AD2

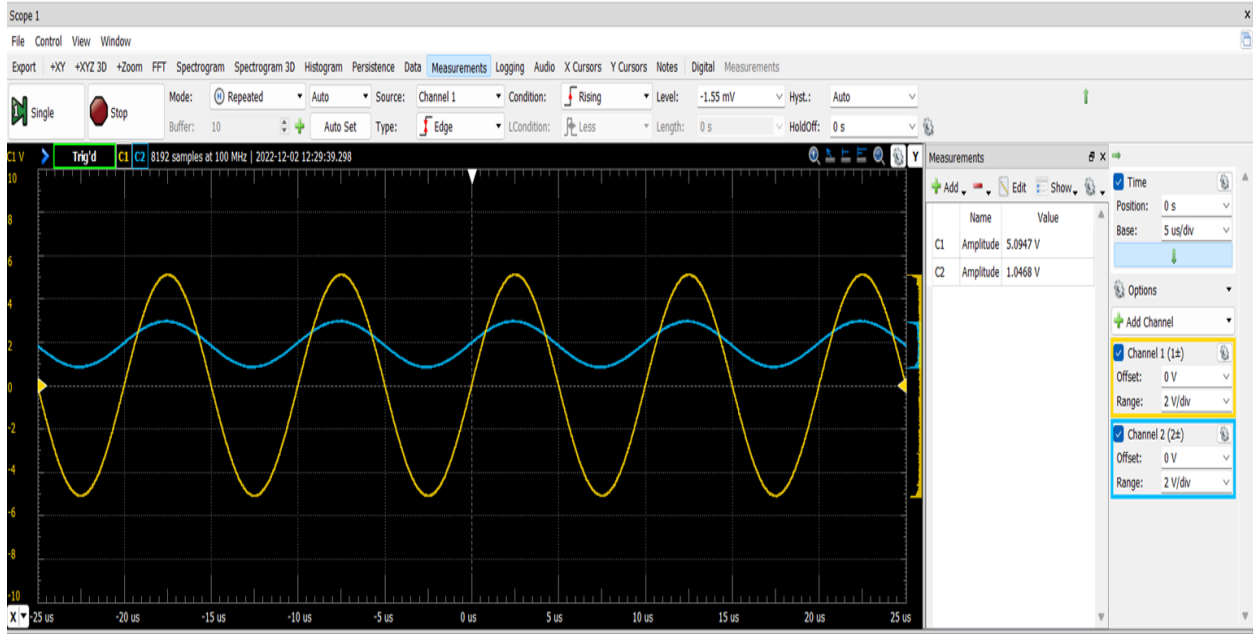


Figure 35: V_{out} vs V_{in} (AD2)

- Gain 1:1

Testing the input with $V_{in} = 5V$

The theoretical result after attenuator: $V_{out} = 5/12.5 = 0.4V$

The theoretical result after gain amplifier: $V_{out} = 0.4 * 1 = 0.4V$

The experiment $V_{out} = 0.396 \text{ V}$

Testing on LTspice

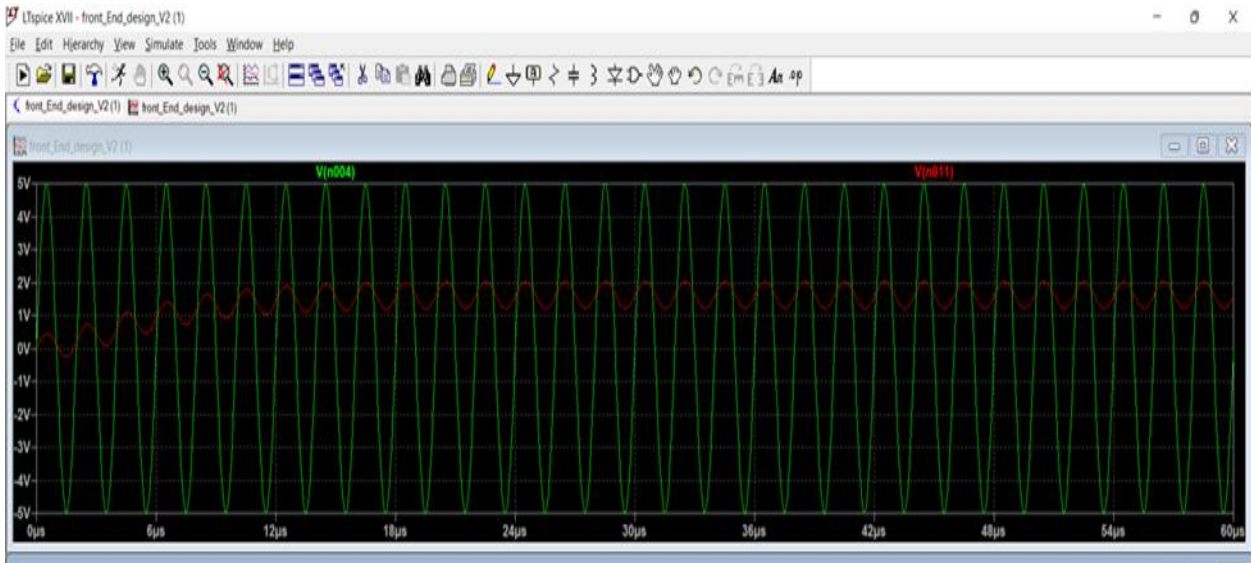


Figure 36: V_{out} vs V_{in} (LT Spice)

Testing on breadboard using AD2

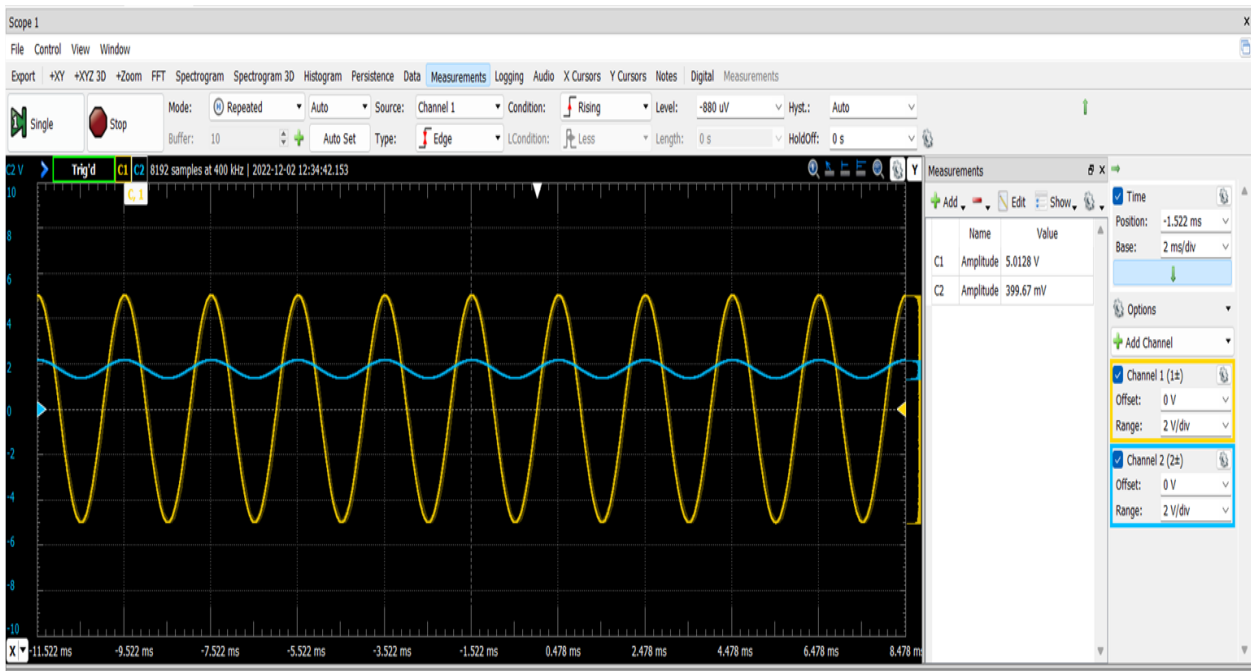


Figure 37: V_{out} vs V_{in} (AD2)

- Bandwidth

Testing on LTspice

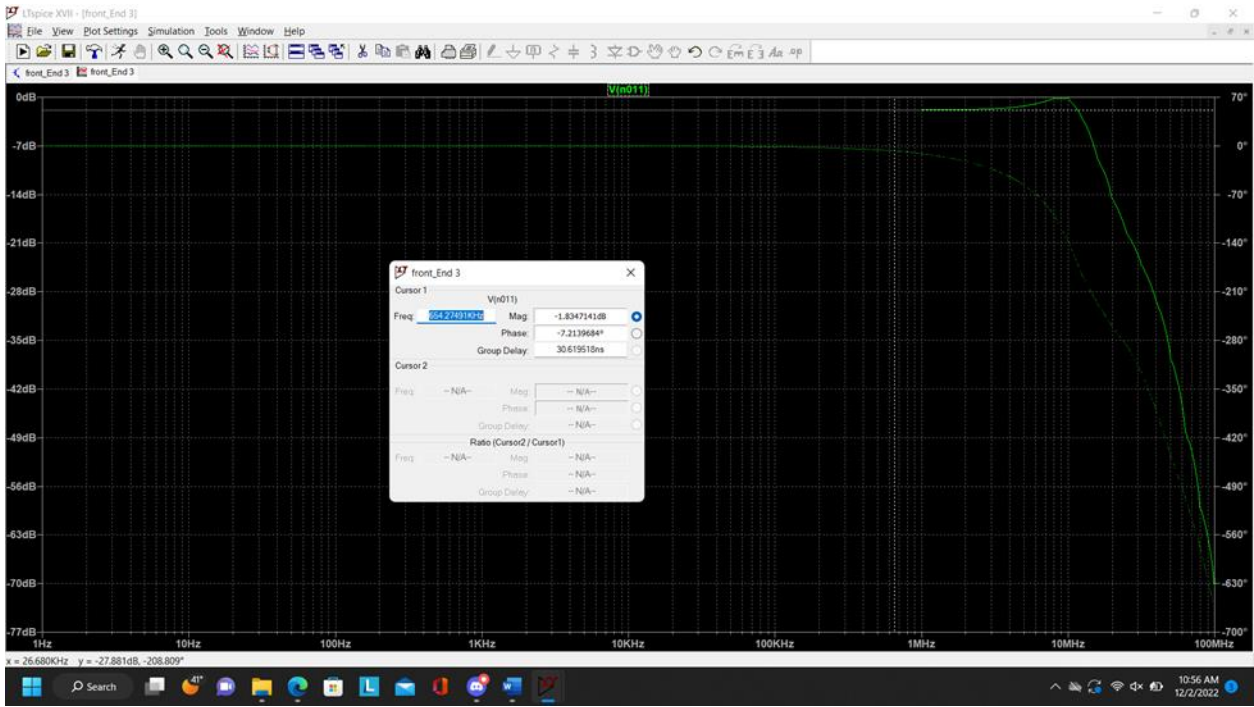


Figure 38: Bandwidth test (LTspice)

Testing on breadboard using AD2

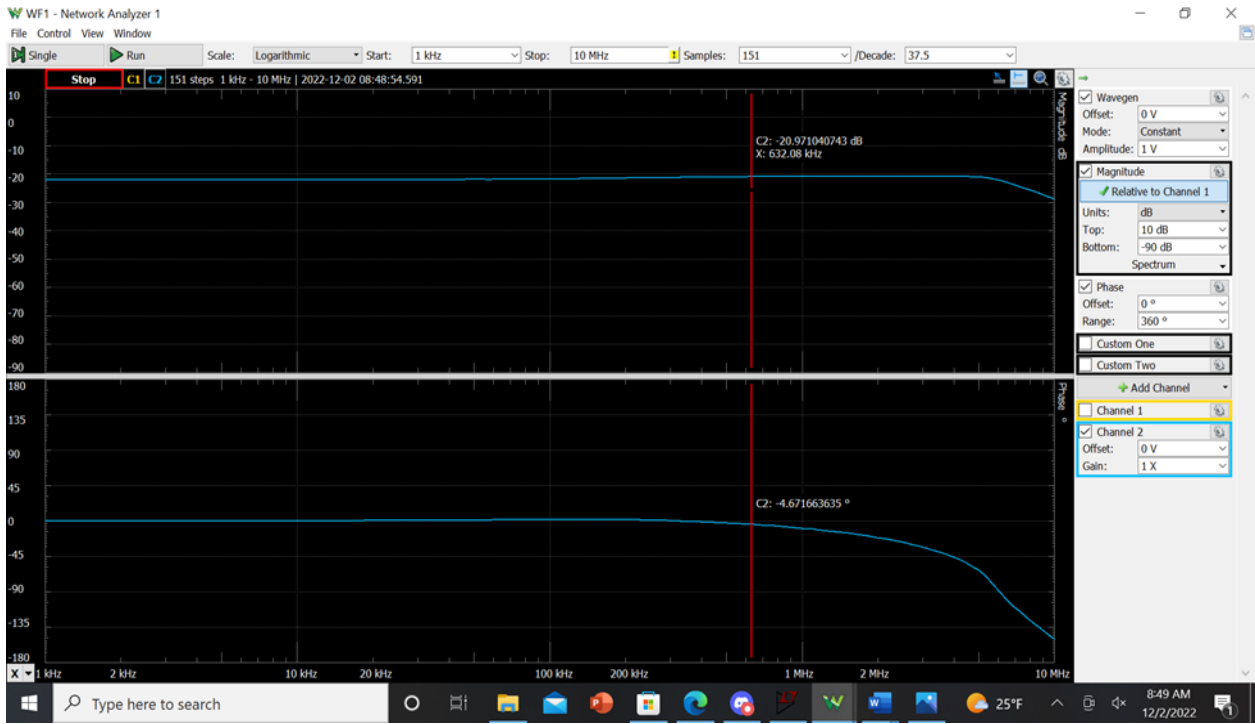


Figure 39: Bandwidth test (LTspice)

6.1.3 USB Soft Start Circuit

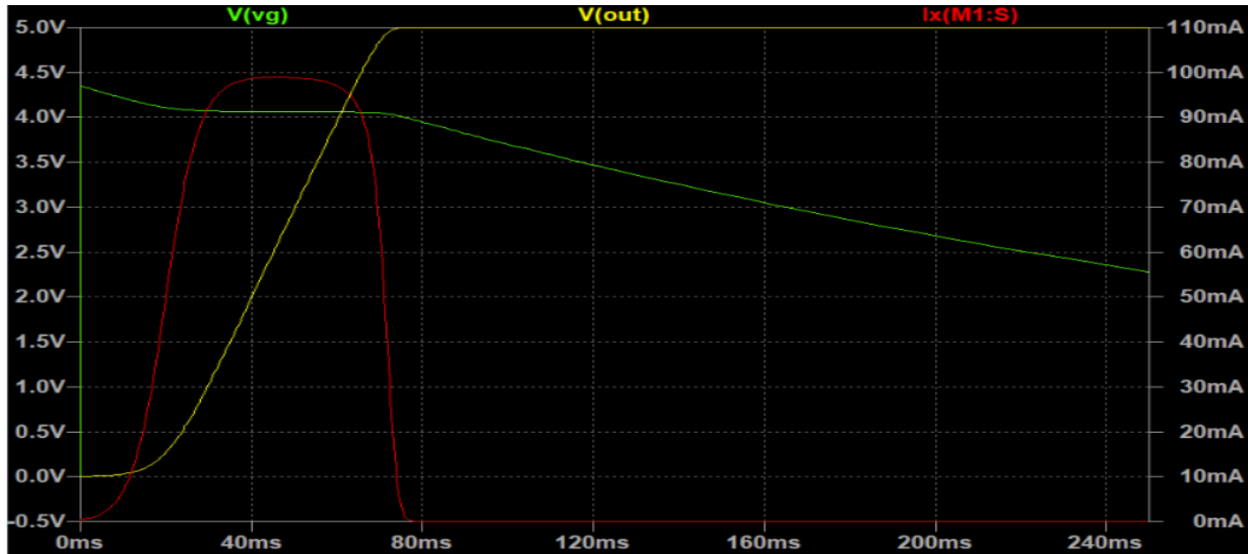


Figure 40: USB Soft Start Circuit Simulation

6.2 MCU Testing

6.2.1 Receive Data from ADC Python Code

The code below is from a computer that receives data from MCU:

```
import time
from tkinter import *
import serial
from serial import Serial
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl

ser = serial.Serial('COM3', 2000000)

#-----
#readline - by default read until end of line
#           with specific number, read until that byte
#read      - by default read 1 byte
#           with specific number, read until that byte
#-----
n = 0
temp = []
while (1):
    data = ser.readline()

    # int(string_number + \r\n), int() eliminates \r\n
    # and convert "string" type -> "integer" type
    print(str(n) + "--" + str(int(data)))
    n = n + 1

    volt = (int(data)*3.3)/4096
    if len(temp)==100:
        plt.plot(temp)
        plt.show()
    else:
        temp.append(volt)
```

The code below is a button that turns on/off ADC:

```

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */
    // counter for depth of buffer and condition (pause) to turn on ADC_DMA
    if (n>=20480 && pause==0){
        n=0;
        HAL_ADC_Start_DMA(&hadc1, (uint32_t*)adc_buff, ADC_BUFF);
    }
    if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13)==0){
        if (pause==0){ // If condition set ADC sampling to 601cycles5
            pause=1; // by add on function changeSampling()
            HAL_ADC_Stop_DMA(&hadc1);
        }
        else{ // Else condition set ADC sampling to original 1cycle5
            pause=0;
            HAL_ADC_Start_DMA(&hadc1, (uint32_t*)adc_buff, ADC_BUFF);
        }
    }
    if(n<20480){
        // copy adc (16 bits) value from buffer to (char) msg
        // this sprintf() COPY adc_buff valaue,
        // CONVERT to "short unsigned" by %hu, PASTE to msg
        sprintf(msg, "%hu\r\n", adc_buff[n]);
        HAL_UART_Transmit(&huart2, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);
        HAL_Delay(1);
    }
    n++; // increasement counter
/* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */

3 /* USER CODE BEGIN 4 */
3 @void HAL_ADC_ConvHalfCpltCallback(ADC_HandleTypeDef* hadc){
0     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, SET);
1 }
2 @void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc){
3     HAL_ADC_Stop_DMA(&hadc1);
4     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, RESET);
5 }
5 /* USER CODE END 4 */
7

```

The code below is changing ADC sampling rate:

```

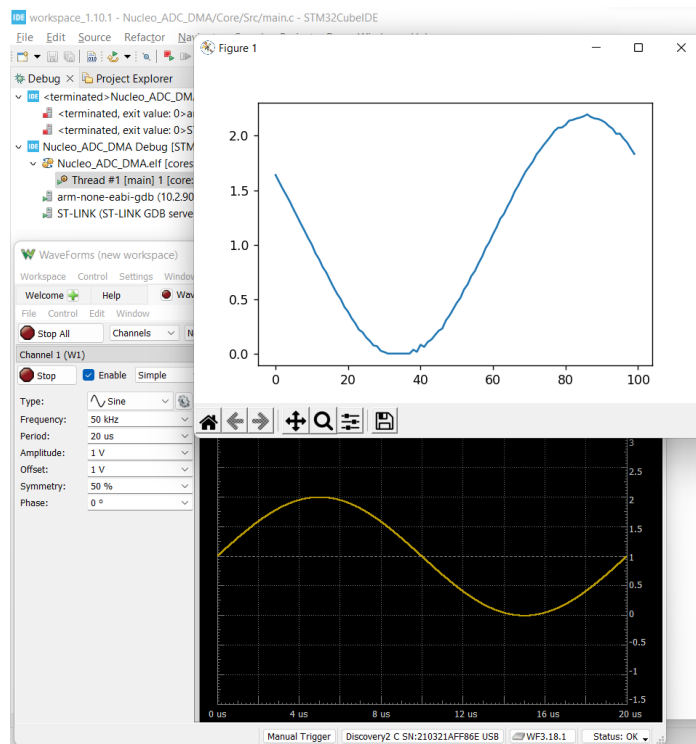
2  /* USER CODE BEGIN WHILE */
3  while (1)
4  {
5      //-----Original-----
6      // turn on ADC_DMA
7      if (n==20480){
8          n=0;
9          HAL_ADC_Start_DMA(&hadc1, (uint32_t*)adc_buff, ADC_BUFF);
10     }
11
12     if (HAL_GPIO_ReadPin(GPIOC, GPIO_PIN_13)==0){
13         if (SMP==0){ // If condition set ADC sampling to 601cycles5
14             SMP=1; // by add on function changeSampling()
15             changeSampling(&hadc1);
16         }
17         else{ // Else condition set ADC sampling to original 1cycle5
18             SMP=0;
19             HAL_ADC_Stop_DMA(&hadc1);
20             LL_ADC_SetChannelSamplingTime(ADC1, LL_ADC_CHANNEL_1,LL_ADC_SAMPLINGTIME_1CYCLE_5);
21             HAL_ADC_Start_DMA(&hadc1, (uint32_t*)adc_buff, ADC_BUFF);
22         }
23     }
24     // copy adc (16 bits) value from buffer to (char) msg
25     // this sprintf() COPY adc_buff yalauae,
26     // CONVERT to "short unsigned" by %hu, PASTE to msg
27     sprintf(msg, "%hu\r\n", adc_buff[n]);
28     HAL_UART_Transmit(&huart2, (uint8_t*)msg, strlen(msg), HAL_MAX_DELAY);
29     HAL_Delay(1);
30     n++;
31
32     /* USER CODE END WHILE */
33
34 /* USER CODE BEGIN 4 */
35 void HAL_ADC_ConvHalfCpltCallback(ADC_HandleTypeDef* hadc){
36     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, SET);
37 }
38 void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc){
39     HAL_ADC_Stop_DMA(&hadc1);
40     HAL_GPIO_WritePin(GPIOA, GPIO_PIN_5, RESET);
41 }
42 void changeSampling(ADC_HandleTypeDef* hadc){
43     HAL_ADC_Stop_DMA(&hadc1);
44     LL_ADC_SetChannelSamplingTime(ADC1, LL_ADC_CHANNEL_1,LL_ADC_SAMPLINGTIME_601CYCLES_5);
45     HAL_ADC_Start_DMA(&hadc1, (uint32_t*)adc_buff, ADC_BUFF);
46 }
47 /* USER CODE END 4 */

```

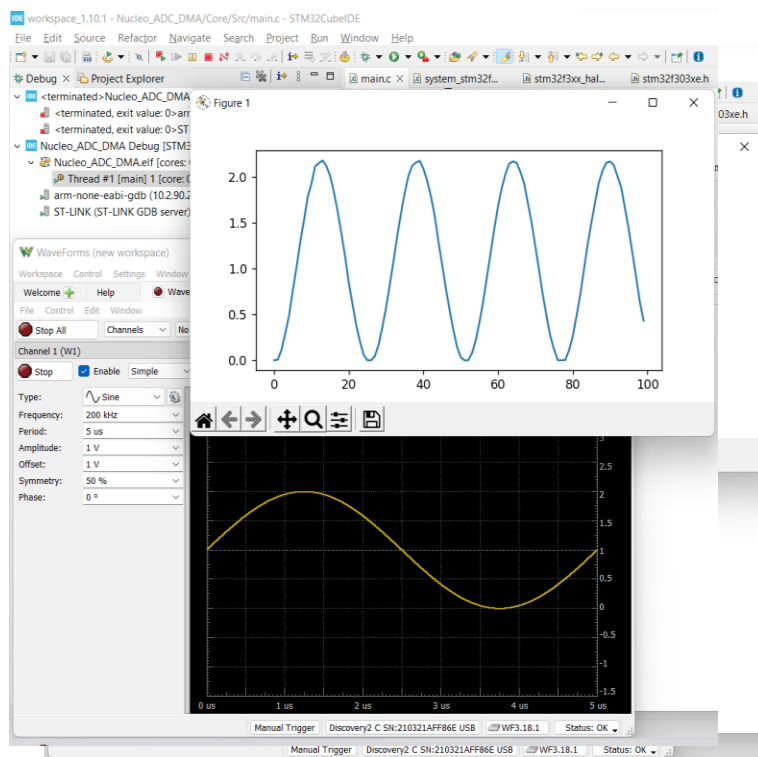
6.2.2 Testing Variable ADC Sampling Rate

- ADC Speed 1 cycle 5

+ Frequency: 50K Hz

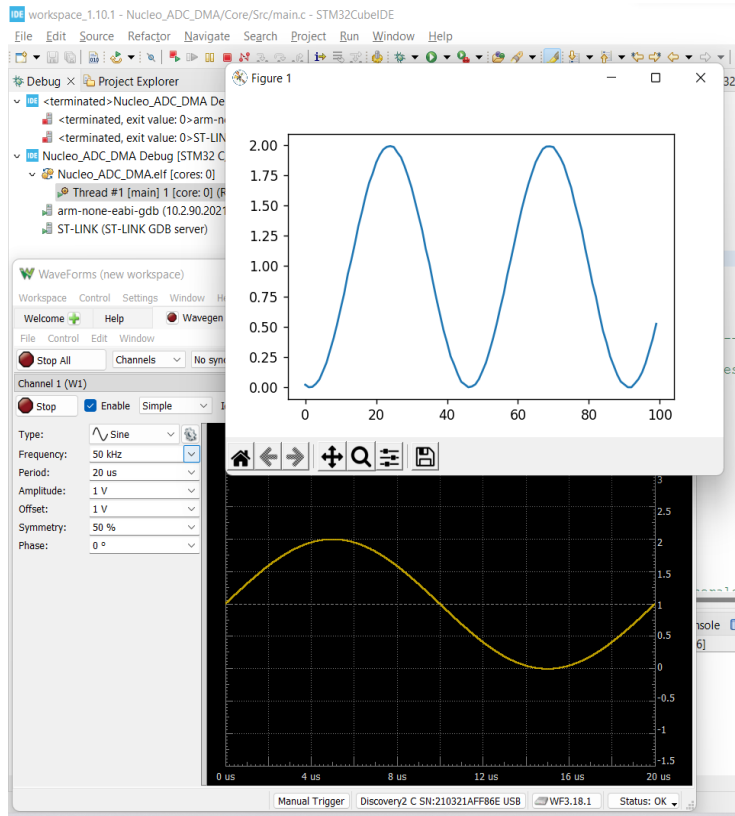


+ Frequency: 200K Hz



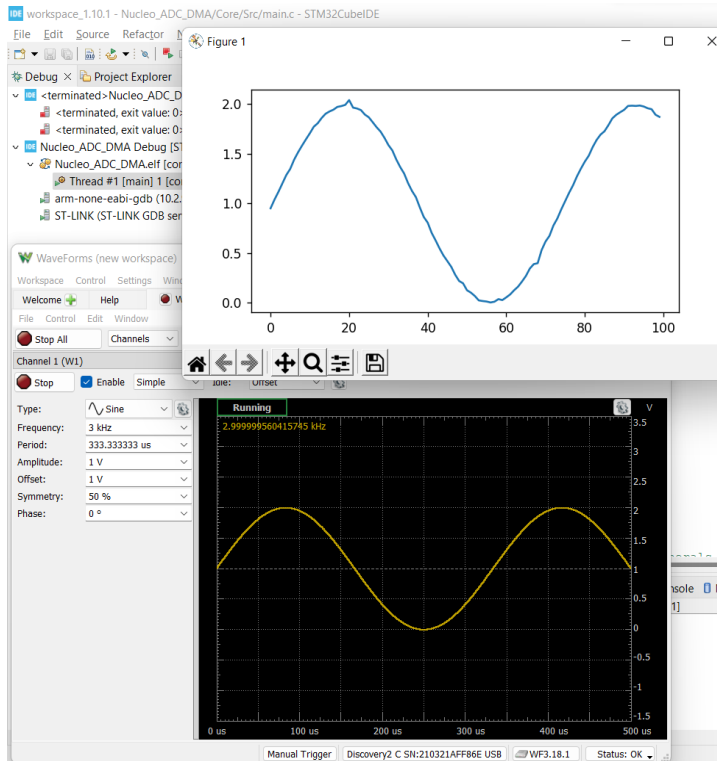
+ Frequency: 500K Hz

- ADC Speed 19 cycle 5
- + Frequency: 50K Hz

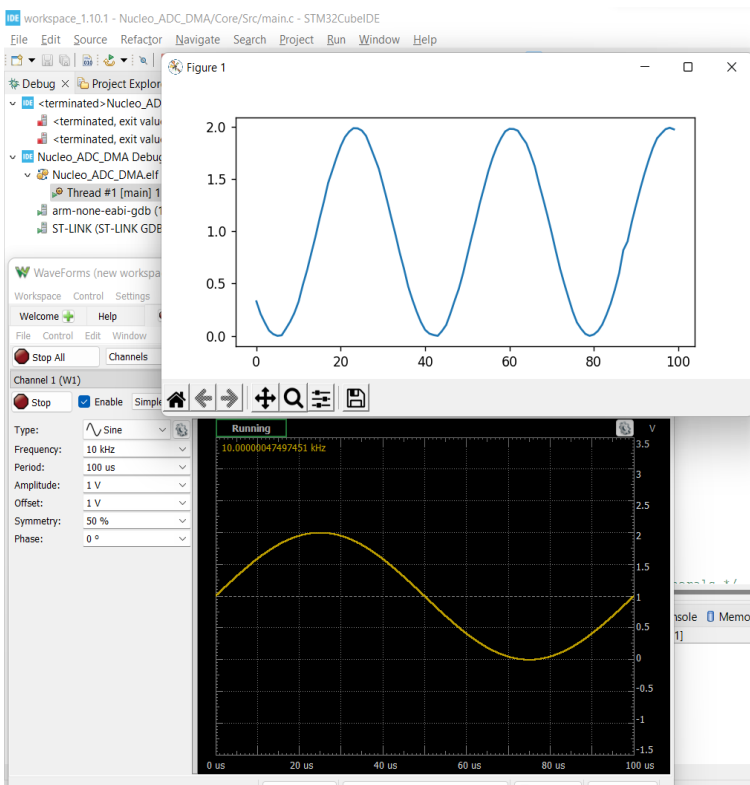


- ADC Speed 181 cycle 5

+ Frequency: 3K Hz

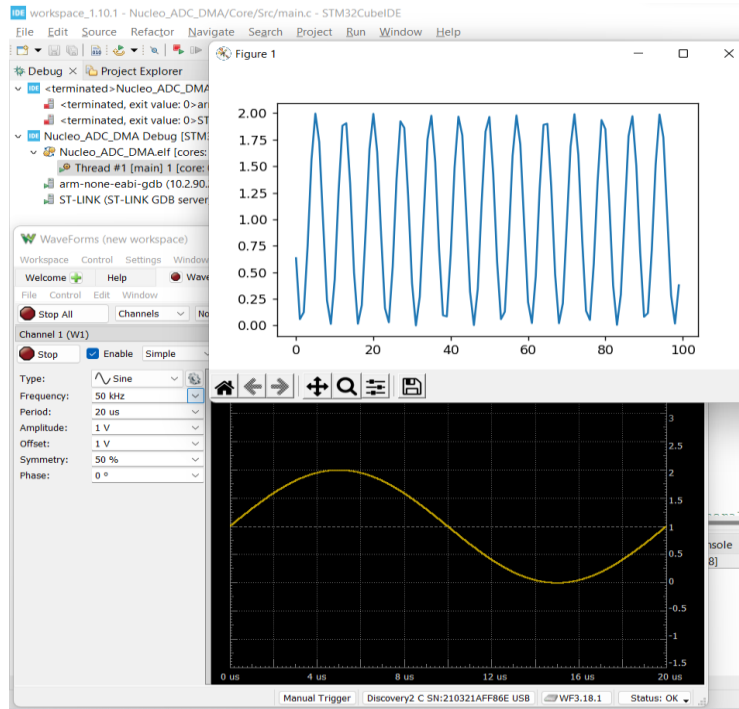


+ Frequency: 10K Hz



+ Frequency: 50K Hz (ADC sampling speed slower compare with high frequency)

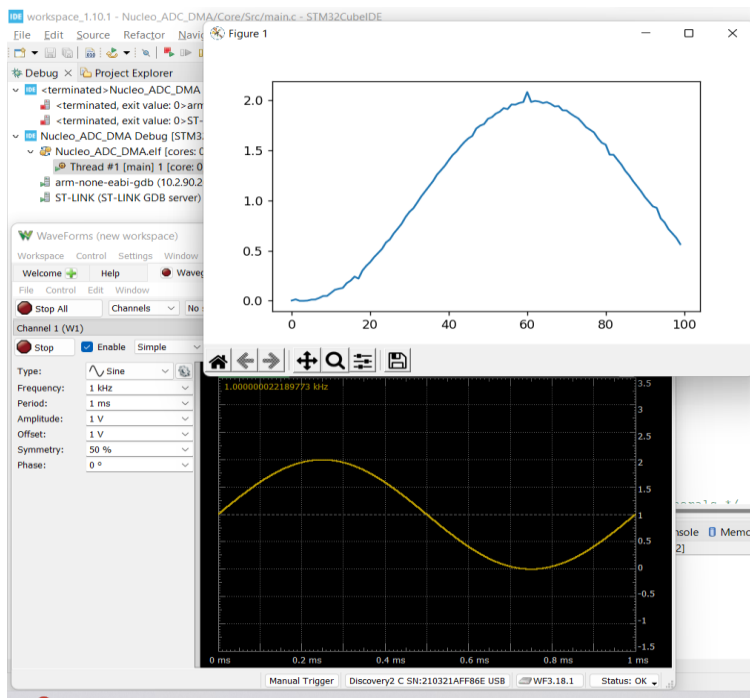
That's why the waveform is incorrect. The data is overwritten.



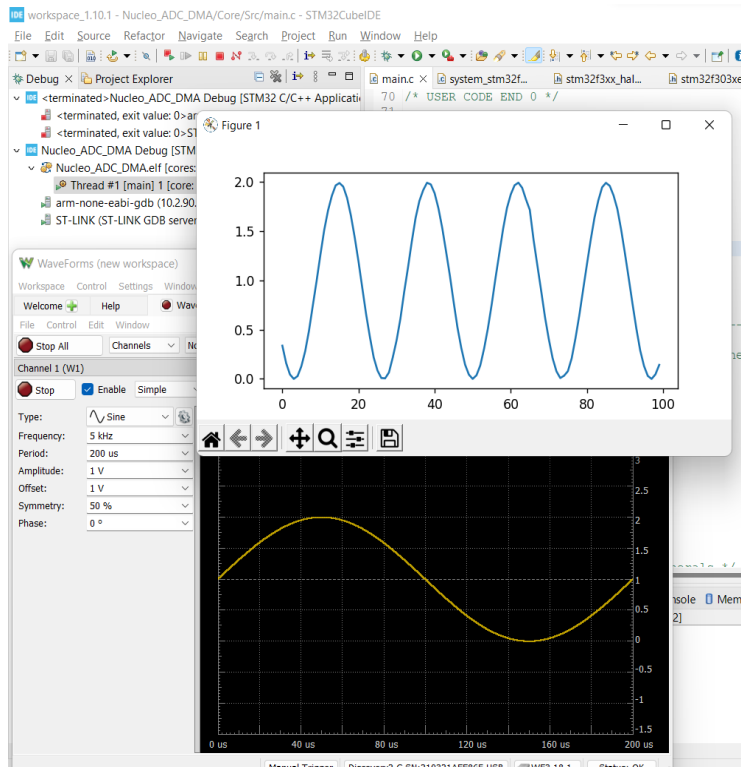
- ADC Speed 601 cycle 5

+ Frequency: 1K Hz (lowest ADC sampling speed faster than 1k Hz or lower frequency)

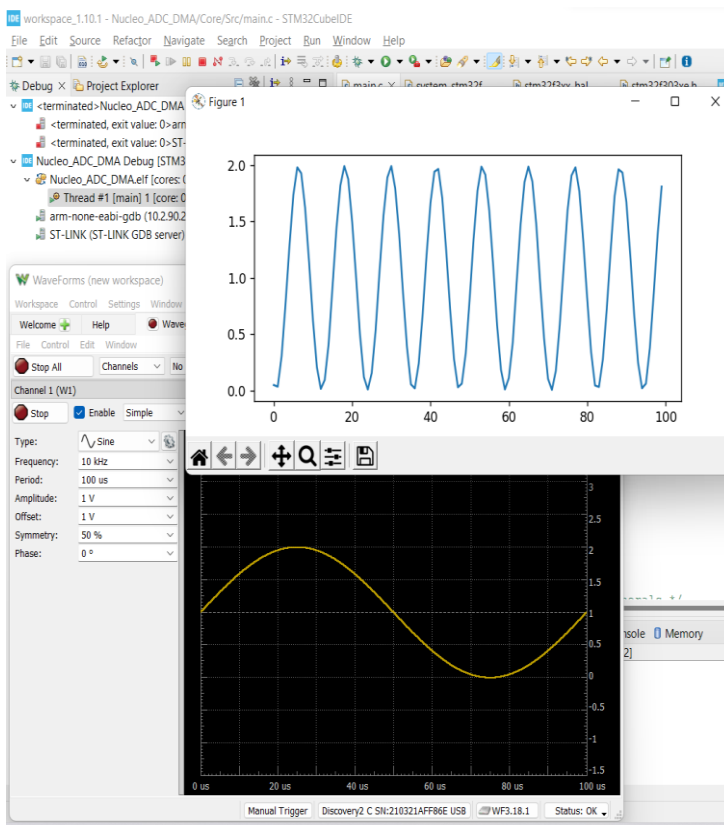
That's why the waveform is not completed within 100 points. Therefore, it needs more than 100 points to complete a waveform.



+ Frequency: 5K Hz

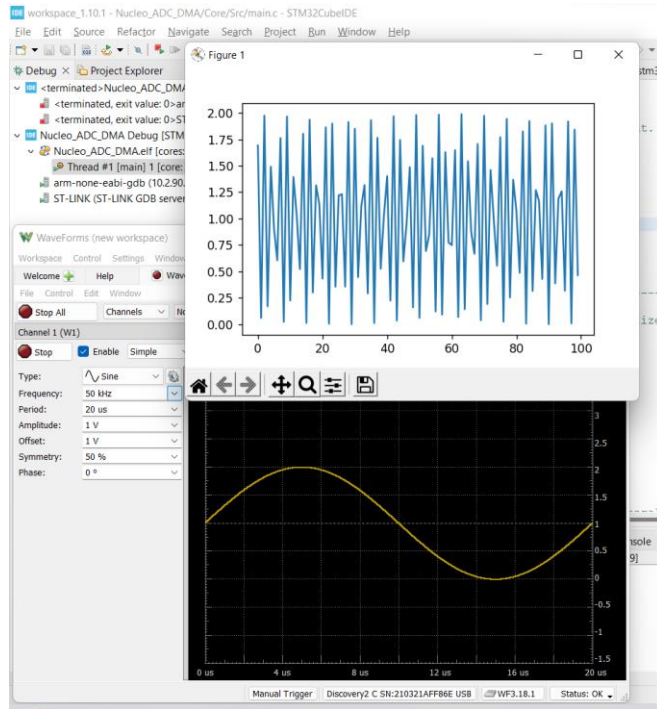


+ Frequency: 10K Hz



+ Frequency: 50K Hz (ADC sampling rate slower than input frequency)

That's why data is overwritten.



6.3 GUI Testing

6.3.1 Speed Testing for PyQtGraph

Speed of PyQtGraph depends on the size of the window screen. On the full size of the window screen, its speed is around 250 fps. That is still fast enough to meet the requirements of the project.

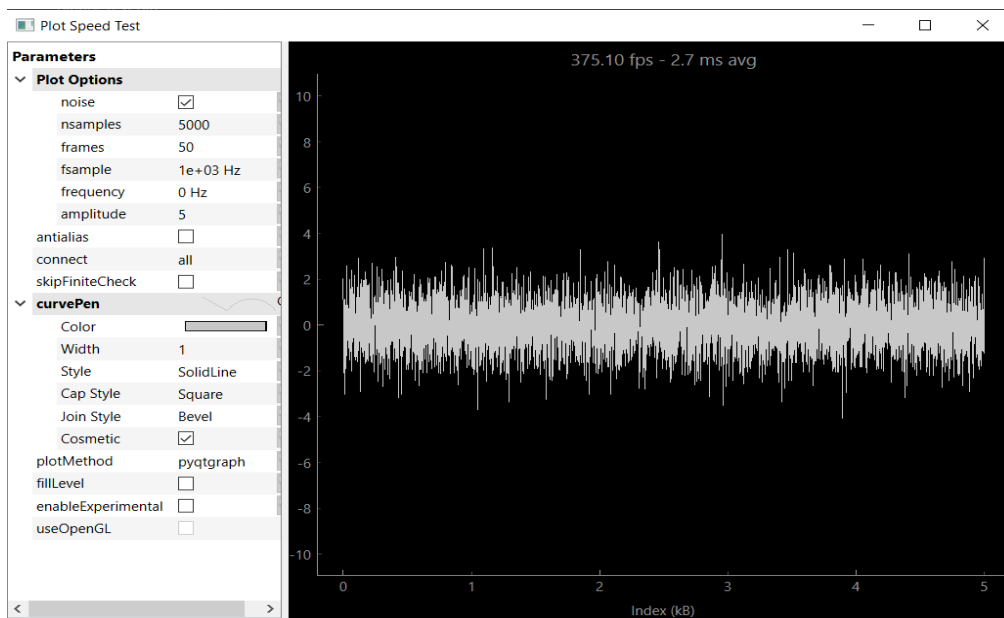


Figure 29: Speed Testing for PyQtGraph

6.3.2 Plot Testing

In this test, we get data from a .bin file that we created and plot waveform in PyQtGraph.

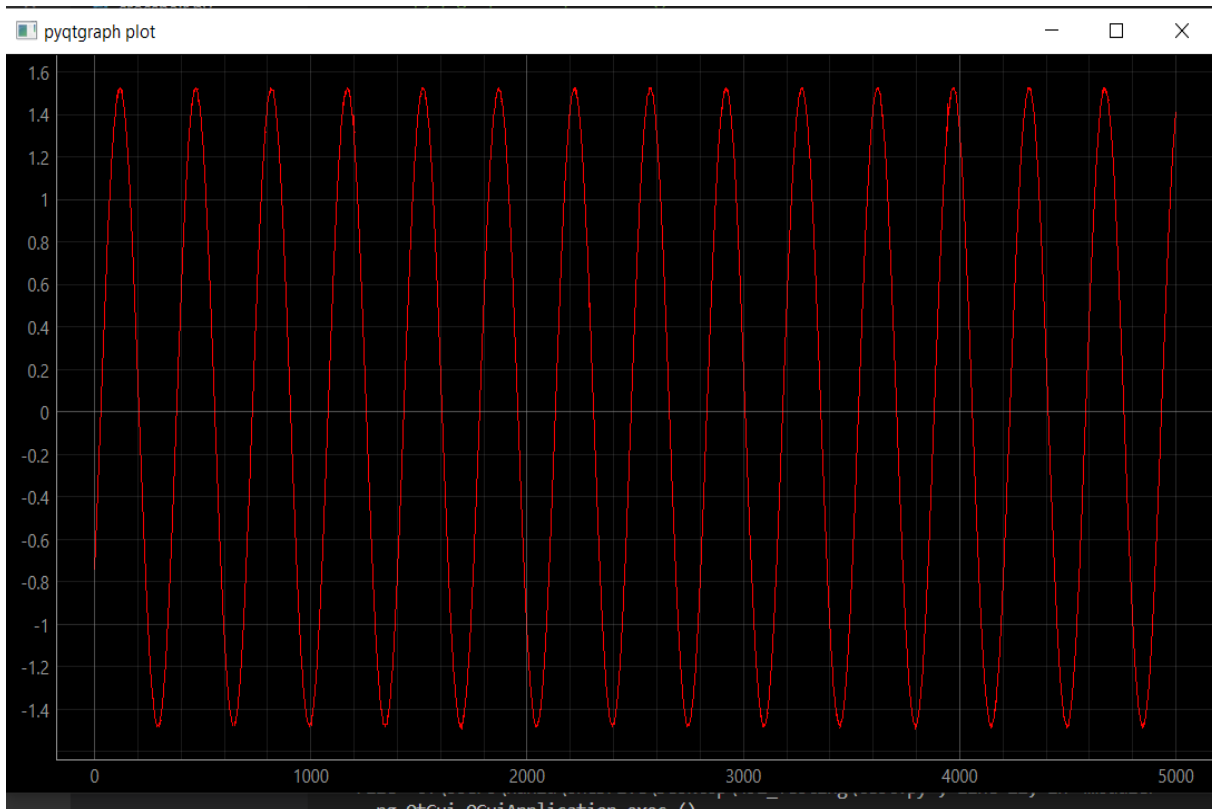


Figure 30: Waveform in PyQtGraph

7. Testing Plan for ECE493

7.1 Analog Front-End Testing

Since the breadboard testing for frequency compensated voltage divider is failed due to the $R1 \cdot C2$ is not equal to $R2C1$ due to the high tolerance of the capacitor and resistor which is 10% and 5% respectively. Therefore, the adjustable capacitor needs to be used in here for fine adjustment in order to get the correct output of frequency compensated voltage divider.

The purpose for this test is to make sure the voltage divider will get the constant value when the frequency varies from 1Hz to 500,000Hz.

7.2 MCU

The goal for this section of the MCU is to start testing and verify different operations in the MCU. Based on the schedule for ECE 493, there are 5 different tests for the MCU.

7.2.1 GPIO Testing

Based on the schedule in section 9, the first testing will be the GPIO test. The GPIO test will include 3 small tests such as attenuator testing, preamplifier testing and DC testing.

7.2.2 Front End - Microcontroller Testing (Breadboard)

In ECE 493, when the front-end completes the design on the breadboard, MCU and front-end will come together for testing to make sure that the front-end can produce the signal and the MCU can receive

it correctly. Also, this testing will include several tests to make sure that the MCU can have a correct control on the front-end.

7.2.3 PCB - GPIO/PWM Testing

The GPIO testing will be the same as section 7.2.1 but with the actual PCB. For the PWM Testing, MPU needs to shift the input signal received from the front-end by using the PWM (DC offset).

7.2.4 PCB - ADC Function Testing

Be able to convert analog to digital with variable sampling speed with the ADC based on input frequency with the PCB. First, we use the STM32cubeIDE tool to set up a high-speed clock, ADC single-ended input mode, and DMA circular mode configuration. Then, we create a 16-bit array buffer and its size is 20480. Then, we use the HAL library to perform ADC and DMA functions. Also, PCB must perform all functions the same as STM32 microcontroller.

7.2.5 PCB - USB Testing (Transmit/Receive)

Communication port USB and Pyserial - transfer/receive data to/from PC, programming language to interact PCB-PC. First, we use the STM32cubeIDE tool to set up USB configuration. Then, we create a message on the Nucleo board and send/receive the message to/from the PC terminal. On the PC terminal, we use Pyserial to communicate with the Nucleo board. When we run both STM32cubeIDE and Pyserial at the same time, they can send/receive messages from each other. Also, the PCB must be able to transfer/receive a signal to/from the PC.

7.3 GUI Testing

The goal for this section of the GUI is to start testing and implementing graphs with live data instead of simulated data through NumPy. USB functionality will be used to get the data from the ADC by using the PyUSB library. After all the bare testing, this section will focus on the creation of the actual interface. The graphical user interface will be made using PyQt and PySide because of their ease of use, wide compatibility with systems, and a Python base. Furthermore, PyQt and PySide are easily integrated into PyQtGraph. The second part of the GUI testing will focus on adding basic oscilloscope functions like voltage per division, signal offset, and time per division. The final stage of testing of the GUI will focus on adding support for more than one input to be shown on the screen. After that, a final sweep of the code will be made to fixup any bugs and do a frequency sweep with two signals at once to make sure that it meets the minimum requirements.

7.4 High-Level Overall System Testing

7.4.1 Calibration

The goal for this test is to find the correction factor of input signal in order to provide the correct input voltage for the ADC. The calibration procedure is to send the known input signal (0-40) Vpp, increase 2V each step, and record the amplitude of signal output. From the result, we will find the correction factor.

7.4.2 Frequency Sweep

The function generator will be used to provide the periodic signal to the device. the frequency from 1 Hz to 500kHz will be swept to test the bandwidth of our device. The same testing will be conducted on another channel for the bandwidth testing.

8. Task Allocations for Remainder of Project

8.1 Analog Front End

Base to the design of the analog front end, here are the remaining task of the project need to be completed by Phu Duong & Giang Nguyen in order to get the project on progress:

- Complete testing the for the invert voltage (-5V)
- Testing the attenuator with the precise resistor and capacitor to prevent to noise of frequency compensated voltage divider
- After the schematic is finalized the PCB design process will be started as soon as possible to allow the high-level test and integration to have time to test and debug any issue that may happen during the project which will be include the following step
 - PCB design in Kicad software.
 - Final check and verify the design
 - Send the design to the manufacturer to print the board.
 - Order all the components that are needed for the board.
 - Older all the components to the board.
 - Conduct the same test on the PCB as in the breadboard test.

8.2 MCU

After being familiar with the microcontroller by using the STM32F303 Nucleo-64 board, the next step is to do several experiments testing of the remaining tasks for the MCU:

- Control the front-end component like VGA by using the switch pin High/Low (FE mux).
- Shift the input signal received from the front-end by using the Timer/PWM (DC offset).
- Be able to convert analog to digital with variable sampling speed with the ADC based on input frequency.
- Testing for GPIO/PWM, ADC function, USB (transmit/receive) with the PCB.
- Examine the calibrating to make sure the device reads data accurately to avoid the device producing false information.

8.3 GUI

These are the remaining tasks that are needed for a successful completion of the project for the GUI part:

- Adding USB support for live data.
- Creating a functioning GUI using PyQt/PySide that imitates a real oscilloscope.
- Adding button functionalities (Voltage/Div; Time/Div; Offset; etc...).
- Adding support for cursor and more than on signal on the screen.
- Frequency sweep with more than one signal to make sure it meets specification.

9. Schedule for Remainder of Project

Base on the given task on section 8 and testing plan need to be conducted on section 7, here are the Gantt chart for the remainder project:



References

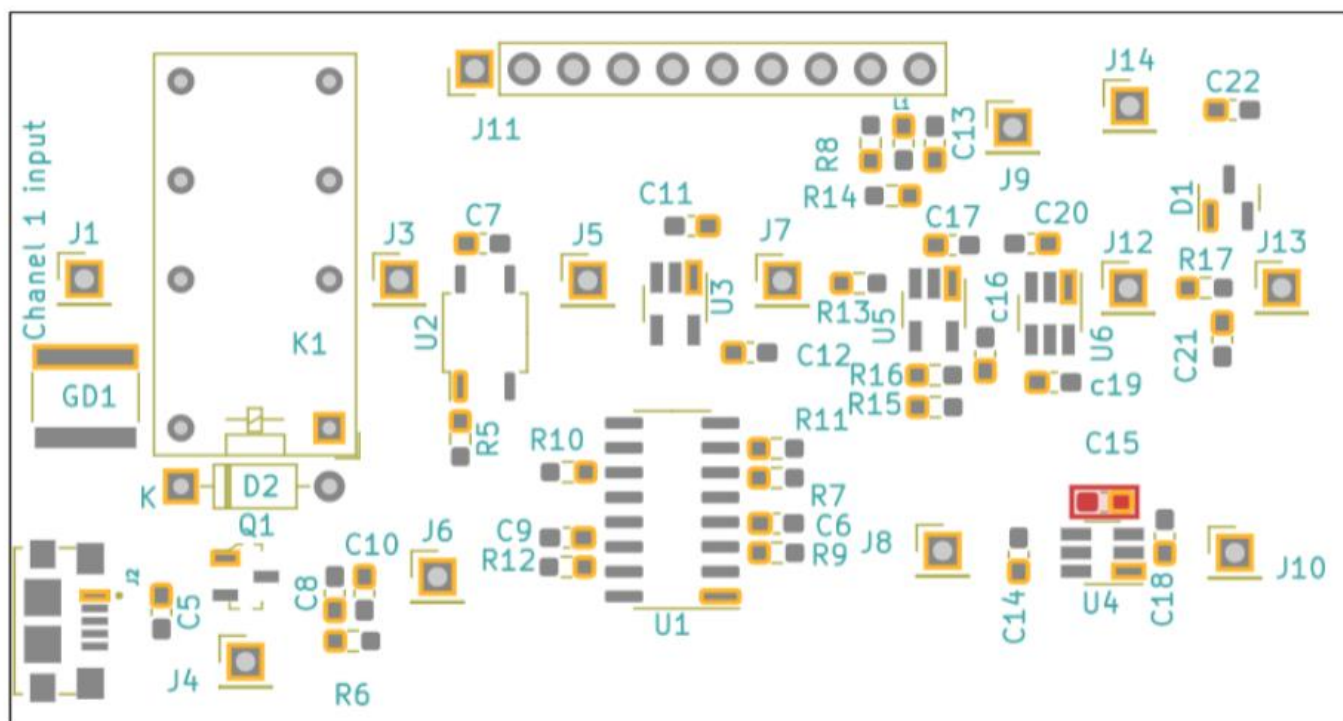
- [1] Mercer, Doug. “Activity: Frequency Compensated Voltage Dividers, For ADALM1000.” *Activity: Frequency Compensated Voltage Dividers, For ADALM1000 [Analog Devices Wiki]*, 22 Aug. 2022, <https://wiki.analog.com/university/courses/alm1k/circuits1/alm-cir-voltage-divider>.
- [2] “AC versus DC Coupling - What's the Difference?” *Siemens DISW*, <https://community.sw.siemens.com/s/article/ac-and-dc-coupling-what-s-the-difference>.
- [3] “Amplifier.” *Wikipedia*, Wikimedia Foundation, 27 Nov. 2022, <https://en.wikipedia.org/wiki/Amplifier>.
- [4] “Buffer Amplifier.” *Wikipedia*, Wikimedia Foundation, 6 Nov. 2022, https://en.wikipedia.org/wiki/Buffer_amplifier.
- [5] Colley, Stephen. “Pulse-Width Modulation (PWM) Timers in Microcontrollers - Technical Articles.” *All About Circuits*, 7 Feb. 2020, <https://www.allaboutcircuits.com/technical-articles/introduction-to-microcontroller-timers-pwm-timers/>.
- [6] Contributor, TechTarget. “What Is Direct Memory Access (DMA)?: Definition from TechTarget.” *WhatIs.com*, TechTarget, 21 Sept. 2005, <https://www.techtarget.com/whatis/definition/Direct-Memory-Access-DMA>.
- [7] Herres, David. “Understanding Sampling Modes in Digital Oscilloscopes.” *Test Measurement Tips*, <https://www.testandmeasurementtips.com/understanding-sampling-modes-in-digital-oscilloscopes/>.
- [8] “STM32CubeIDE.” *STMicroelectronics*, <https://www.st.com/en/development-tools/stm32cubeide.html>.
- [9] *STMicroelectronics*. https://www.st.com/resource/en/user_manual/um1725-description-of-stm32f4-hal-and-lowlayer-drivers-stmicroelectronics.pdf.
- [10] OzyOzk. “OzyOzk/Oguplot: Real Time Python Plot.” *GitHub*, <https://github.com/OzyOzk/Oguplot>.
- [11] “PyQtGraph#.” *PyQtGraph*, <https://pyqtgraph.readthedocs.io/en/latest/index.html>.
- [12] “Scientific Graphics and GUI Library for Python.” *PyQtGraph*, <https://www.pyqtgraph.org/>.
- [13] Suyash458. “SUYASH458/SoftwareOscilloscope: A Software Oscilloscope for Arduino Made with Python and PyQtgraph.” *GitHub*, <https://github.com/Suyash458/SoftwareOscilloscope>.

13. Appendix C: BOM for single channel Front End Design

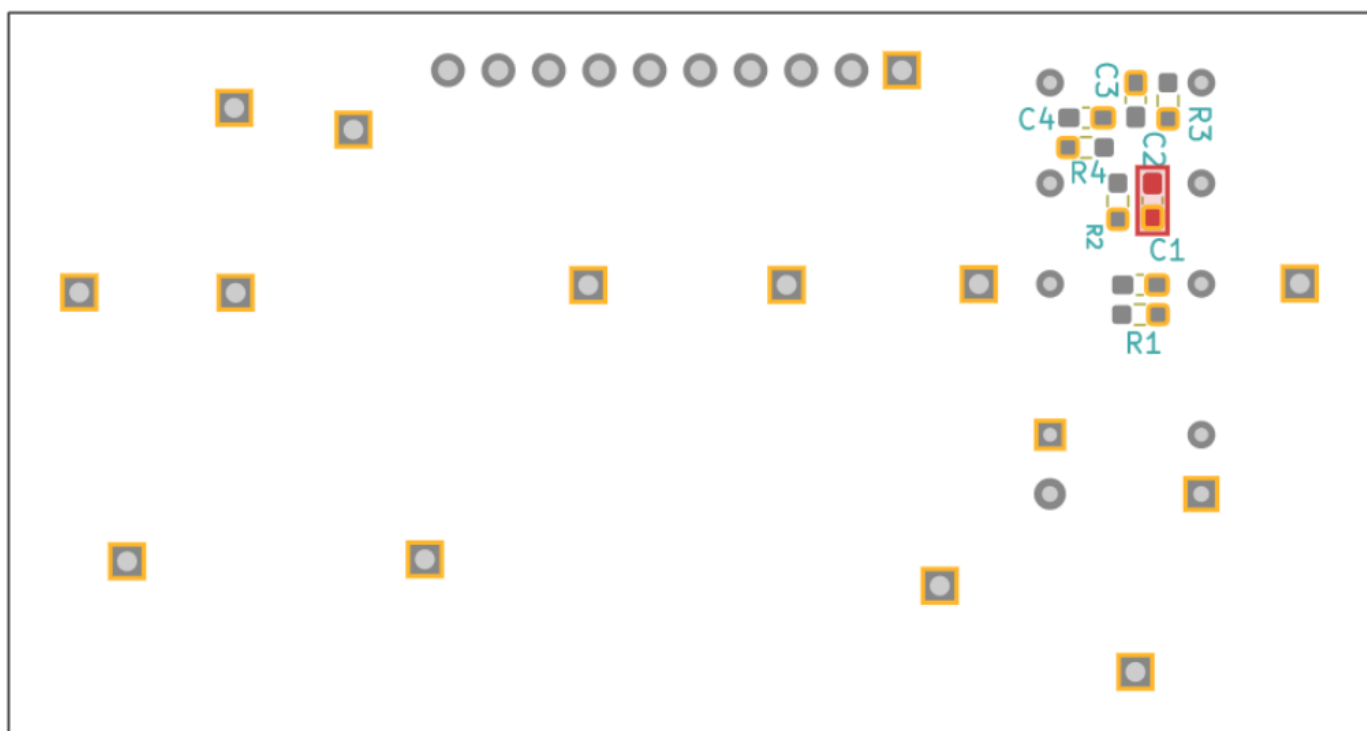
BOM for single channel Front End Design			
Number	References	Value	Quantity
1	C6, C9, C11, C12, c16, C17, c19, C20, C22	100nF	9
2	C5, C14, C18	10uF	3
3	C2, C4	820pF	2
4	C13, C15	1uF	2
5	C1	200pF	1
6	C3	68pF	1
7	C7	47nF	1
8	C8	1.5uF	1
9	C10	1mF	1
10	C21	160pF	1
11	R7, R13, R16	1K Ohm	3
12	R1, R3	820KOhm	2
13	R8, R9	6.2 Ohm	2
14	R14, R15	2kOhm	2
15	R2	200kOhm	1

16	R4	68KOhm	1
17	R5	240 ohm	1
18	R6	27kOhm	1
19	R10	3.9KOhm	1
20	R11	1.5KOhm	1
21	R12	9.1KOhm	1
22	R17	100 ohm	1
23	L1	10uH	1
24	D1	BAV199DW	1
25	D2	1N4148	1
26	U3, U5	LMP7731	2
27	U1	DG412xY	1
28	U2	AQY284SX	1
29	U4	LTC1983ES6-5#TRPBF	1
30	U6	OPA863	1
31	GD1	GDT_SH90	1
32	K1	G5V-2	1
33	Q1	IRLML6402TRPBF	1

34	J4, J8	Conn_01x01	2
35	J1	Channel 1 input	1
36	J2	2174507-2	1
37	J3	T_att	1
38	J5	T_acdc	1
39	J6	T_P5V	1
40	J7	T_amp	1
41	J9	T_pwm	1
42	J10	T_N5V	1
43	J11	GPIO	1
44	J12	T_buf	1
45	J13	ADC	1
46	J14	+3.3 V	1



Front PCB component for Front End single channel layout



Back PCB component for Front End single channel layout