March 17, 2023
Letter of Transmittal

Dr. Mike Hegedus
School of Engineering Science,
Simon Fraser University,
Burnaby, B.C.
V5A 1S6

RE: ENSC 405W Design Specification

Dear Dr. Hegedus,

Please find attached a copy of a design specification document prepared by SyncLock for ENSC 405W/440.

The DLA will allow users to extract a signal from a noisy environment and will be designed on a PCB with a digital-to-analog converter, a microprocessor, and an analog-to-digital converter. These components will receive an input signal and process it to extract a low-noise signal from the input. The system will allow parameters such as amplitude, phase, and frequency to be used to configure the device to allow it to be used for testing different applications in different environments.

This document will cover the design requirements which include hardware, digital, software, and mechanical components. We also provide an appendix stating the design alternatives and testing plan to ensure the device works as intended. Finally, we will outline the details for the proof of concept stage when we demo our project in April.

Thank you for taking the time to review our design specification document for the digital lock-in amplifier. Should any questions or concerns arise, please do not hesitate to contact our Chief Communication Officer, Ese Dan-Aighewi, by email (adanaigh@sfu.ca).

Sincerely,

Ese Dan-Aighewi
Chief Communication Officer
SyncLock

Digital Lock-In Amplifier

Design Specification appendix

SyncLock (Company #8)

Simon Fraser University
ENSC 405W - Capstone A

March 17, 2023

Sponsor:
Intelligent Sensing Laboratory
Supervisors:
Dr. Behraad Bahreyni, PEng
Dr. Fatemeh Es.haghi

Members:
Ese Dan-Aighewi
Minghui Liang
Brayden McKeen
Lucien Somorai
Yupeng Zhao
Haoran Zhou

Abstract

This document outlines the design aspects of the digital lock-in amplifier device for both the proof of concept and prototype phases. The specifications will cover hardware, digital, software and mechanical information regarding design. We will also justify all the choices we make stating the reasoning behind them. Finally, we will conclude with an alternative design plan as well as a testing plan going forward.

Table of Contents

List of Figures

List of Tables

List of Equations

Glossary

Term	Definition
ADC	Analog-to-Digital Converter
DAC	Digital-to-Analog Converter
DC	Direct Current. A signal which exhibits either positive or negative values
DDS	Direct Digital Synthesis, to generate reference signal
DLA	Digital Lock-In Amplifier
DSP	Digital Signal Processing
FIR Filter	Finite Impulse Response, used to implement low pass filter
GPIO	General Purpose Input/Output pins, to set voltage rate and connect external peripherals.
MATLAB	Simulation and computational software platform
MCR	Matlab Compiler Runtime, a set of libraries used to compile and execute MATLAB applications
Microcontroller	A small computer to control the components of the device
PCB	Printed Circuit Board
Preamp	Preamplifier, amplifies input signal for processing by ADC
Raspberry Pi	Single board mini-computer, running on Linux
RG	Gain Resistor, control the gain of instrumentation amplifier
SPI	Serial Peripheral Interface, a serial protocol for connecting external peripherals
SSH	Secure Shell, to connect to Raspberry Pi terminal

Version History

Design Specifications Document Version	Date
1	March 7, 2023
2	March 12, 2023
3	March 15, 2023
4	March 16, 2023
5	March 17, 2023

Approvals

This document has been reviewed and revised by SyncLock providing the design specifications currently in development for the lock-in amplifier.

1 Introduction

In optics, microelectronics, nanotechnology, and sensing areas, extracting small signals buried in noise plays a very important role. It is essential to have a device that can inform researchers the signals generated are what they want. To make small signals measurable, traditional amplifiers can't do the job because they will amplify noise as well. To solve this issue, SyncLock company is working on a portable digital lock-in amplifier (DLA) that can not only extract buried small signals accurately but also is more configurable than analog lock-in amplifiers. Figure 1 below pictures the flowchart and a simple block diagram showing the steps to operate our device and the simplified process of extracting small signals. A more detailed block diagram will be discussed later in the document.

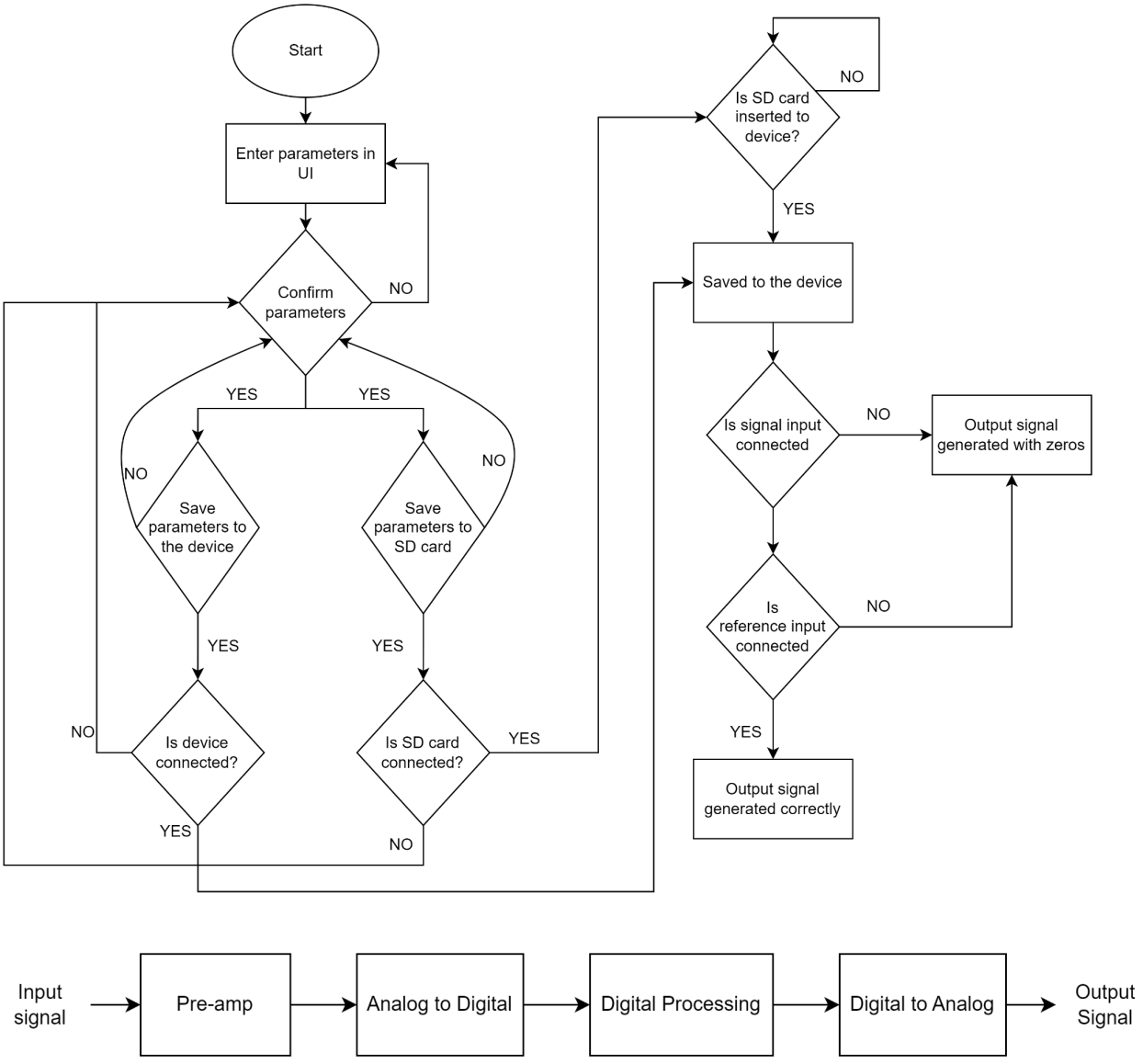

Figure 1, DLA flowchart and block diagram of the simplified process.

1.1 Scope

This document outlines the device design in detail and explains the approaches made by SyncLock company toward the design. The design consists of hardware, digital, software, and mechanical components. At the end of the document, we also included two appendices that detail our design alternatives and test plans.

1.2 Challenges

Listed here are the current technical challenges in designing the proof-of-concept stage of the DLA. This list doesn't include all challenges.

- Bandwidth issue of the pre-amplifier chip
- Compatibility issue between components and Raspberry Pi
- Processing speed issue with Raspberry Pi
- Potential non-linearity in analog-to-digital converter (ADC)

1.3 Updates on Feedbacks

This project is conducted by Professor Bahreyni and the post-doctoral researcher, Fatemeh. SyncLock company is receiving feedback from them on a weekly basis. Besides that, SyncLock company also has progress meetings with Professor Hegedus and Usman to receive feedback on current progress. Below is the list of updates we made by considering the feedback.

- Buy components with lower sampling rates to get hands on the system instead of spending too much time on component selection.
- Buy an appropriate case and use a gasket to fix the device in the case instead of 3D printing the casing at the proof-of-concept stage
- The poster demo needs to show the audience how noisy the DLA can extract signals, so the audience can have a better understanding.

1.4 Product Stage Classification

Same as the requirement document, the project stages are classified into three categories listed below.

Tag	**Project stage**
A	Proof-of-concept
B	Engineering Prototype
C	Production Version

Table 1. Project stage convention.

2 System Overview

The DLA device workflow consists of five different components: pre-amplifier, ADC, processor, DAC, and DDS. And three other components to support them: a power supply, SD card reader, and USB port. Figure 2 illustrates a detailed block diagram of the DLA. In the figure, the input signal is amplified to a detectable range for ADC. Then ADC converts the analog signal to the digital signal for digital processing. In the processor, the digital signal is processed through a series of mixing and filtering to get its in-phase and quadrature components. Combining both components, the desired digital signal has been extracted. Later, the DAC component takes the processed digital signal into the analog domain for output. The “lock-in” term in DLA requires a reference signal to extract the small signal buried in noise. In our design, users are given the option to use an external reference signal. Alternatively, users can choose an internal reference signal generated by the DDS component and connect the “reference output” to the “reference input” through a cable. This adds more flexibility to our device. On the other hand, the power supply provides power to all components, the SD reader can save data into an SD card and the USB port allows the device to communicate with a computer.

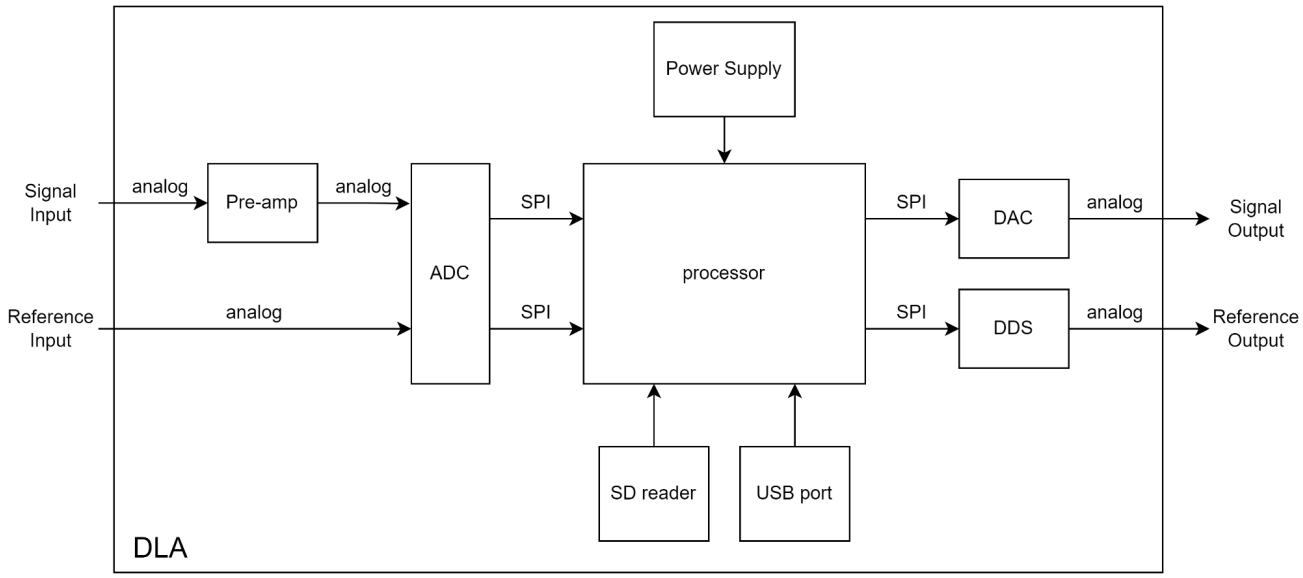

Figure 2. Detailed block diagram.

In the proof-of-concept stage, a Raspberry Pi is used as the processor for simplicity. It has its own power supply, SD reader, and USB port, so more focus can be put on the important components. The pre-amplifier, ADC, DAC, and DDS will be mounted on a breadboard and connected to the Raspberry Pi to show the overall system is working.

In the engineering prototype stage, we will select a suitable microcontroller for the processor and all components will be mounted on a PCB. In addition, the components mentioned above, mounted on the PCB will also be an SD card holder and an array of connectors to interface with each of the enclosure-mounted ports, power switch, batteries, and LED. The power supply will be replaced by rechargeable batteries for portability.

In the final production version, the company will design a unique casing to contain and protect the PCB.

3 Design

This section provides the design specifications for different aspects of the system overall.

3.1 Hardware Design Specifications

Table 2 below provides the hardware specifications and their associated requirements.

ID	Tag	Description	Changes	Req ID
	A	Two different ADCs have been ordered.	Changed R 2.3.4 to 10-bit resolution due to shipping and cost concerns	R 2.3.2 R 2.3.3 R 2.3.4
	A	DAC has been ordered	Changed R 2.3.6 to 10-bit resolution due to shipping and cost concerns	R 2.3.2 R 2.3.6 R 2.3.9 R 2.3.10
	A	Preamp is purchase and will be placed according to the block diagram	None	Req 2.3.5
	A	Unity Gain-Bandwidth of Preamp need to over 1.5MHz	None	R 2.3.15
	B	The Gain of Preamp can be controlled	None	R 2.3.16
	B	Power is provided through rechargeable batteries	None	C 3.1.6

Table 2. Hardware design specifications and related requirements.

3.1.1 Preamp

The preamp is a component that amplifies the amplitude of the input signal to an appropriate size so that the full scale of the ADC can be taken advantage of. It has two key aspects, one is its bandwidth and the other is its noise. Its bandwidth needs to exceed the value of the input signal frequency multiplied by the gain required by the input signal so that the input signal will not be distorted when it passes through the preamp. At the core of the preamp circuit

will be an instrumentation amplifier, chosen for its desirable noise characteristics over other op-amp designs.

Proof-of-concept

In this project, we will buy the existing instrumentation amplifier, which is better than constructing it by the regular op-amp. The gain of the instrumentation amplifier depends on the value of the resistor RG, which is connected between the RG pins. Figure 12 of section 3.1.2 shows how the instrumentation amplifier will be wired with the ADC and Raspberry Pi. In Figure 12, RG has a value of 12.4 kΩ providing an example gain of 5 V/V. In stage A, we will adjust the gain of the instrumentation amplifier by changing the resistor RG directly.

We will purchase a few instrumentation amplifiers that meet our device requirements to test them. Their key information is listed in Table 3, pin functions are listed in Table 4, and their simplified internal schematic and their pin configurations are listed in Figure 3-8 [1, 2, 3].

Component Model	Bandwidth	Noise	Gain	Output Voltage
INA823DGKT	1.9 MHz(G = 1) 60 kHz(G = 100)	21nV/√Hz	$1 + \frac{100k\Omega}{R_G}$	$V_{output} = G(V_{+in} - V_{-in}) + V_{Ref}$ Equation 1: Output Voltage Calculation of preamp
INA821IDRGR	4.7 MHz(G = 1) 290 kHz(G = 100)	7nV/√Hz	$1 + \frac{49.4k\Omega}{R_G}$	
INA819ID	2 MHz(G = 1) 270 kHz(G =100)	8nV/√Hz	$1 + \frac{50k\Omega}{R_G}$	

Table 3. Component key information.

Pin	DESCRIPTION
-IN	Negative (inverting) input
+IN	Positive (non-inverting) input
OUT	Output
RG	Gain setting pin. Place a gain resistor between RG pins
REF	Reference input. This pin must be driven by a low impedance source
-VS	Negative supply
+VS	Positive supply

Table 4. Pin function.

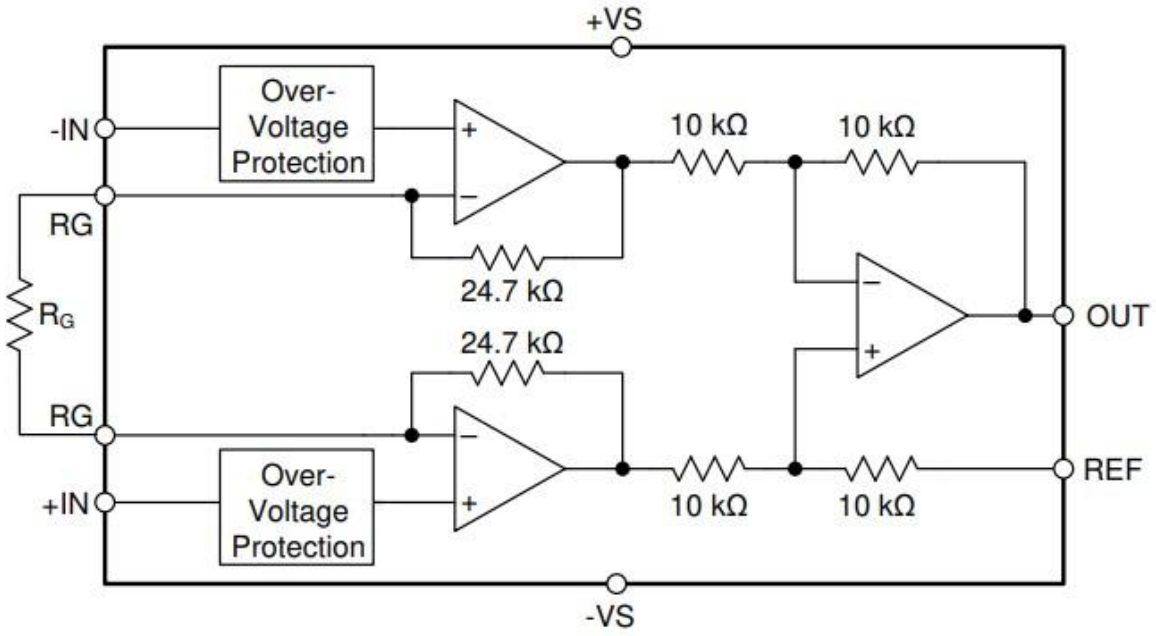

Figure 3. INA821 Simplified Internal Schematic.

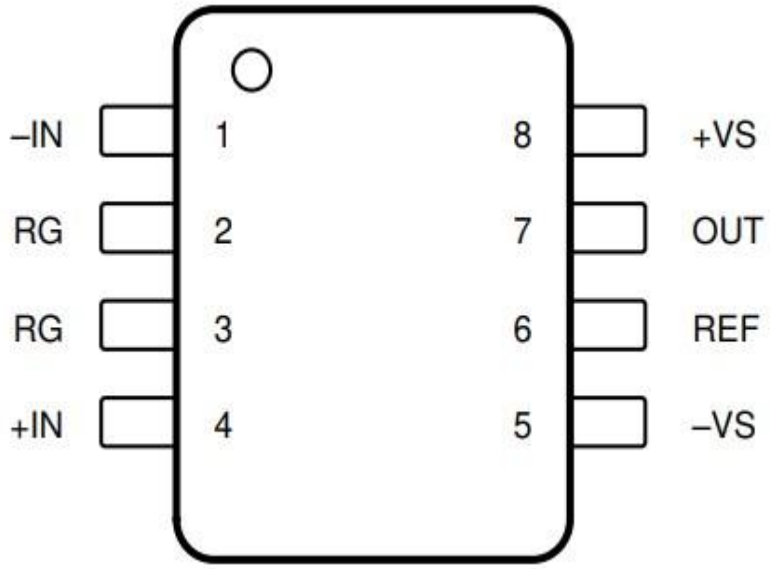

Figure 4. INA821 8-Pin Configuration Top View.

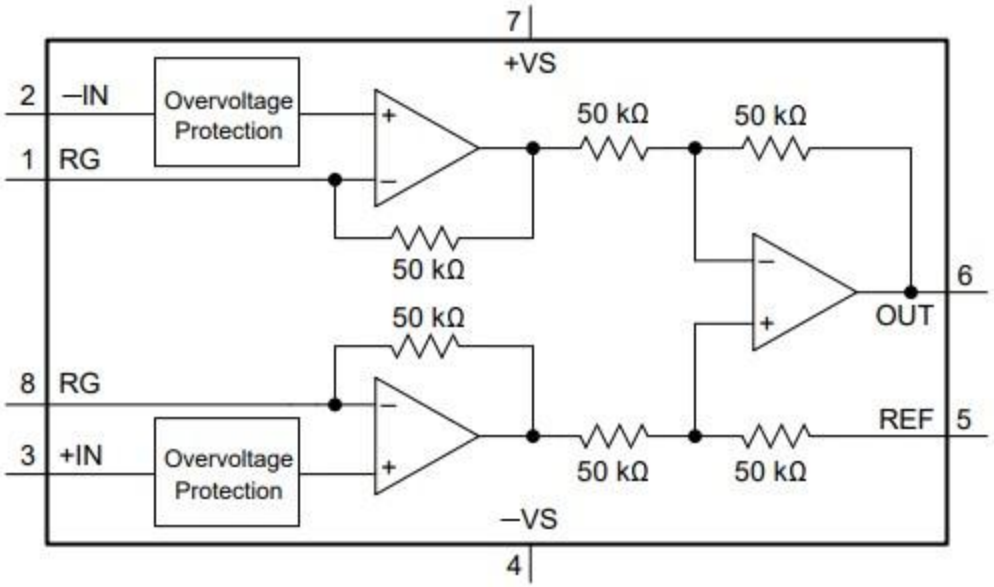

Figure 5. INA823 Simplified Internal Schematic.

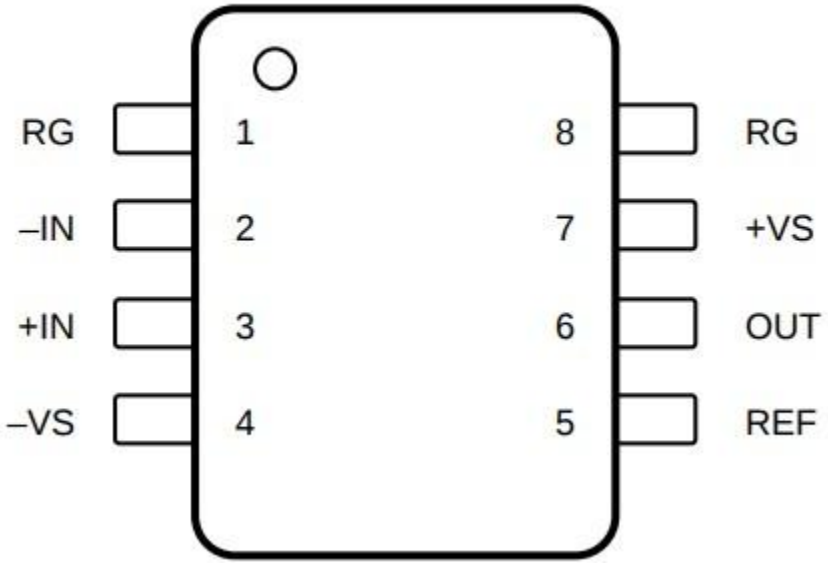

Figure 6. INA823 8-Pin Configuration Top View.

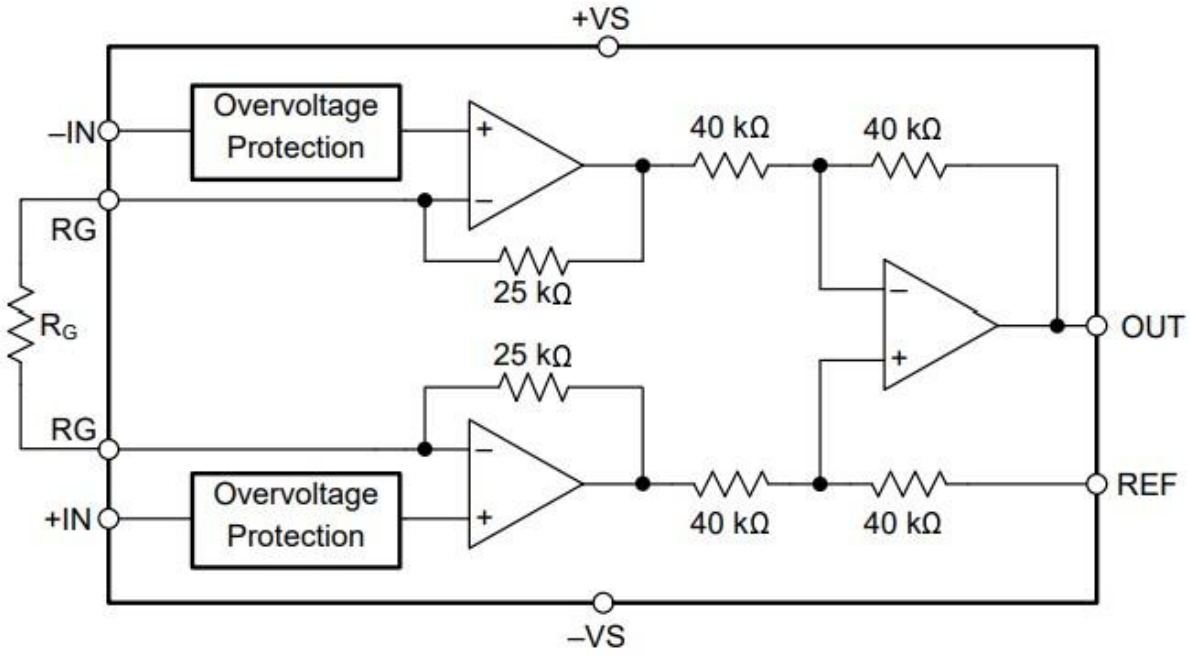

Figure 7. INA819 Simplified Internal Schematic.

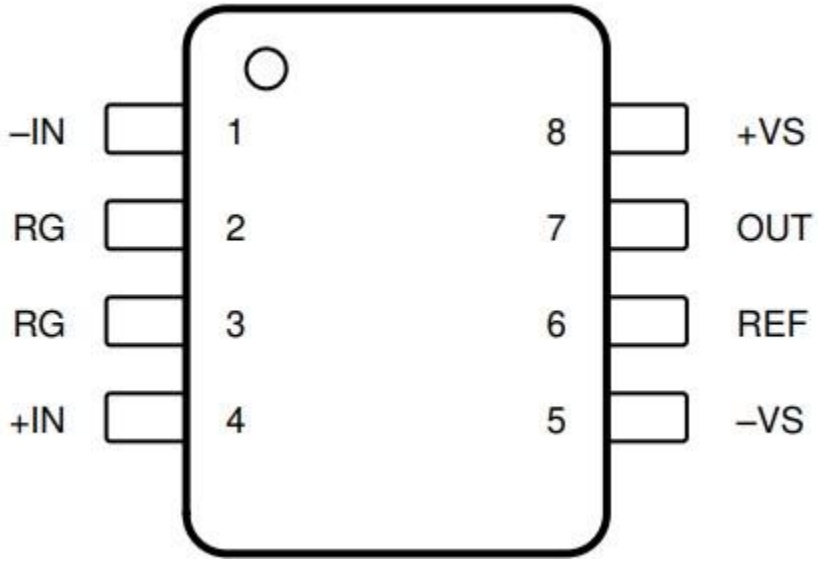

Figure 8. INA819 8-Pin Configuration Top View.

Prototype

In stage B, we will design a controlled gain circuit in the preamp section, which can control the gain of the amplifier by a microprocessor. It is composed of a voltage-controlled amplifier circuit

and a control circuit. The amplifier circuit is similar to stage A and its gain can be controlled by the external voltage input. The control circuit will receive the digital signal from the microprocessor and feed it into the voltage control amplifier circuit.

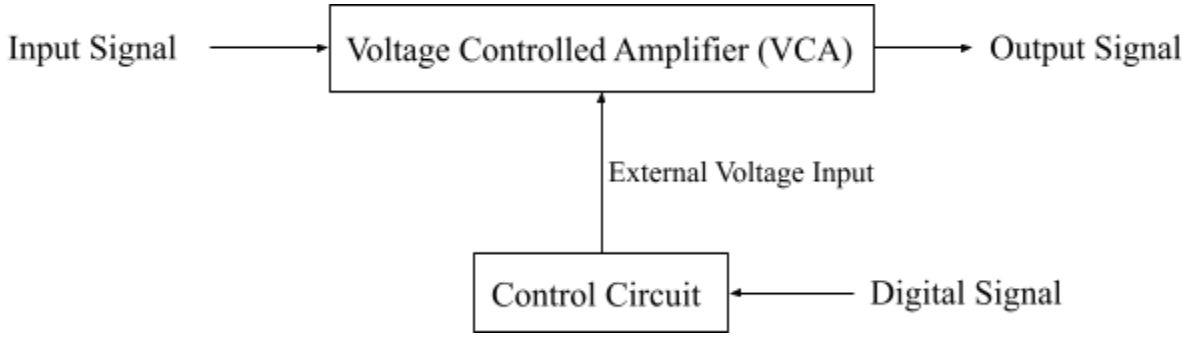

Figure 9. Controlled Gain Circuit Block Diagram.

3.1.2 Analog to Digital Converter (ADC)

An ADC is a device that converts analog signals to digital signals. The lock-in amplifier utilizes an ADC to convert input analog signals to the digital domain. There are two key aspects of an ADC: sampling rate and resolution. Sampling rate is the rate at which it samples the analog signal. The Nyquist rate is the minimum sampling rate required to characterize a signal and that is two times larger than the frequency of the analog signal. Resolution is the smallest step the ADC can detect, larger resolution means the digital signal after conversion is more accurate.

Proof-of-concept

In this project, we will buy existing commercial ADC products for simplicity and time-saving. In stage A, we will purchase an inexpensive ADC with a low sampling rate and a low resolution.

Below are the two ADCs we ordered: MCP 3008 and ADS1115.
The MCP 3008 has 8 multiplexed input channels, 10-bit resolution, and a sampling rate of 200 kHz. ADS1115 has 2 input channels, 16-bit resolution, and a sampling rate of 860 Hz. Both ADCs support SPI communications and can be powered by 5V. This means that, for stage A, an ADC can be mounted on a breadboard and connected to a Raspberry Pi.

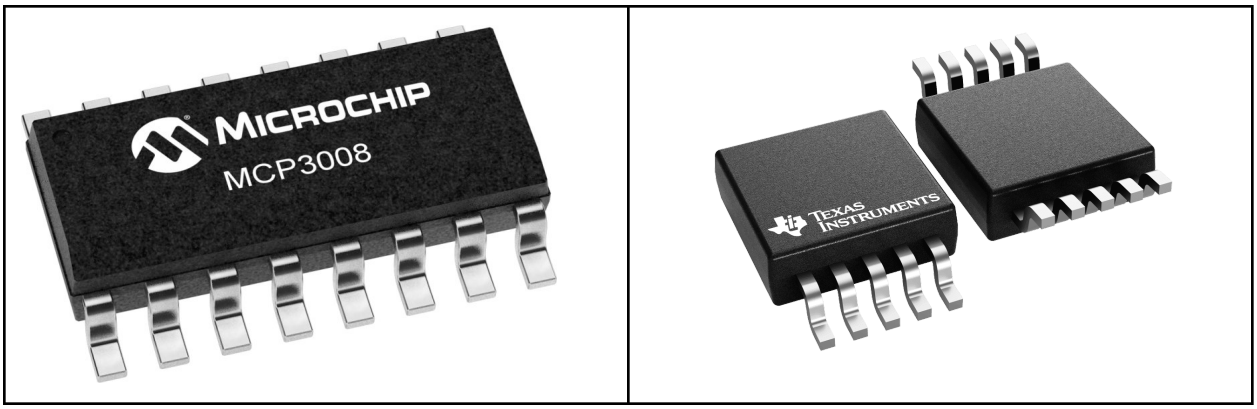

Figure 10. MCP 3008 (left) and ADS1115 (right) ADCs [4, 5].

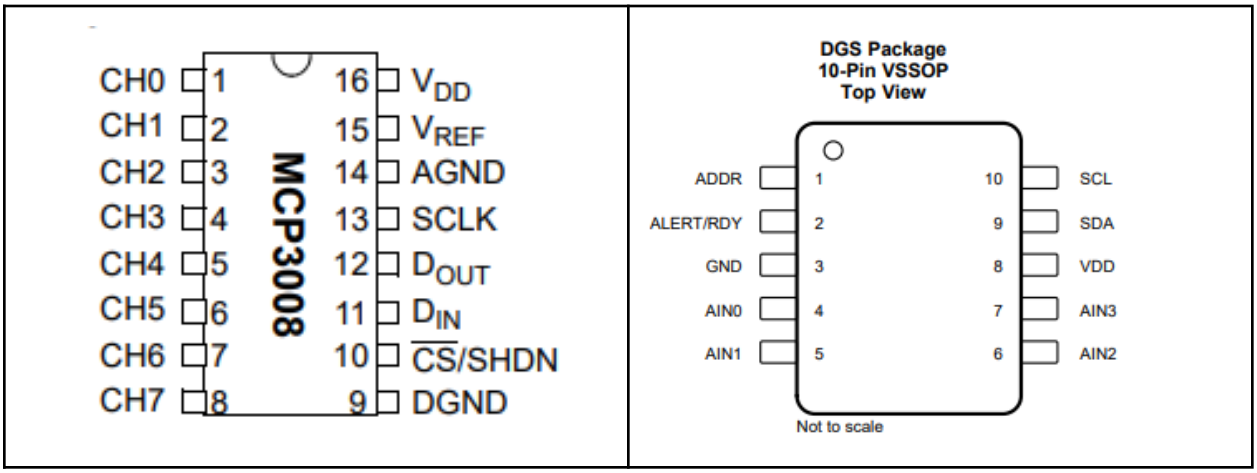

Figure 11, Pin configuration of MCP 3008 (left) and ADS1115 (right) [4, 5].

Below is a schematic showing how the INA821 instrumentation amplifier, MCP3008 ADC, and Raspberry Pi will be wired for the proof of concept.

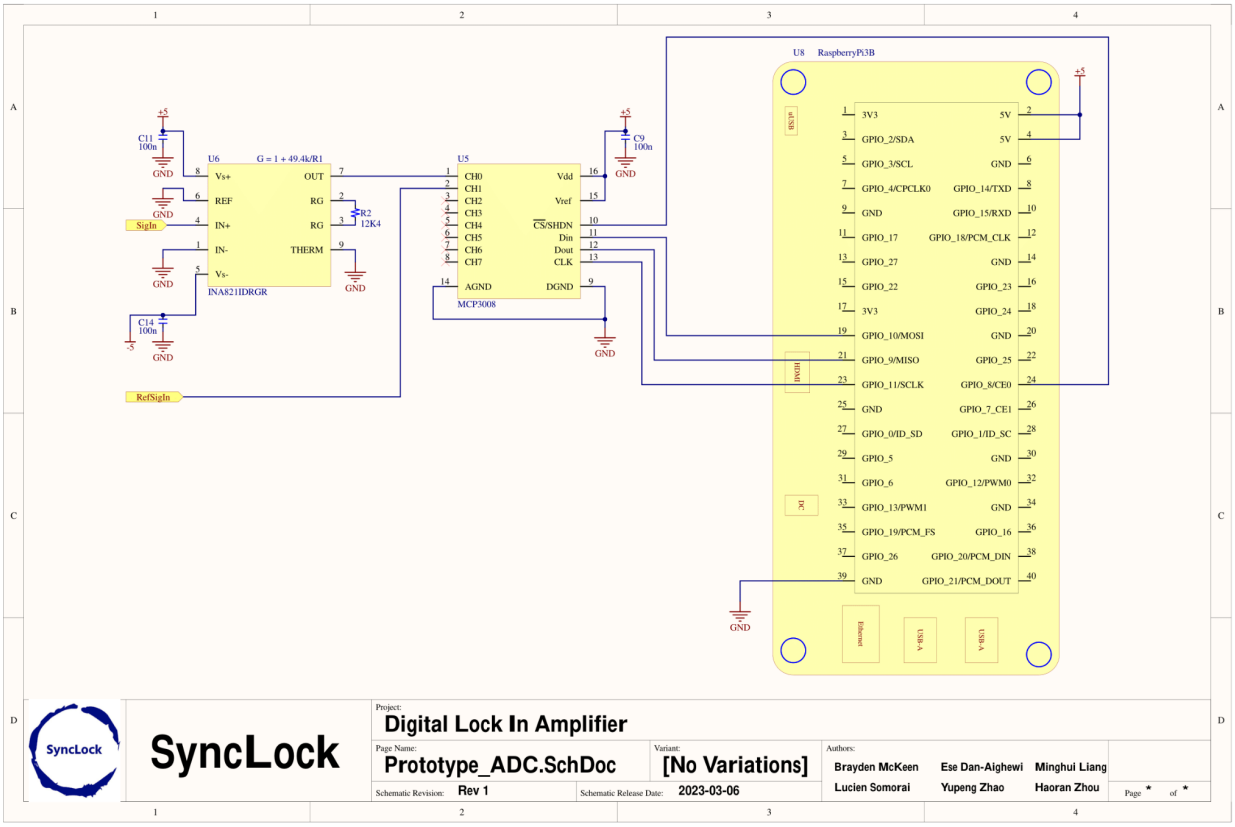

Figure 12. Schematic of MCP 3008, INA821, and Raspberry Pi 3 Model B.

Prototype

In stage B, an ADC that fully satisfies the requirements mentioned in the requirement document will be selected. Our current choice is the AD7134BCPZ-RL7 which has 4 input channels, 24-bit resolution, and a sampling rate of 1.5 MHZ. This ADC meets all ADC requirements as we mentioned in the requirement document. This selection is subject to change.

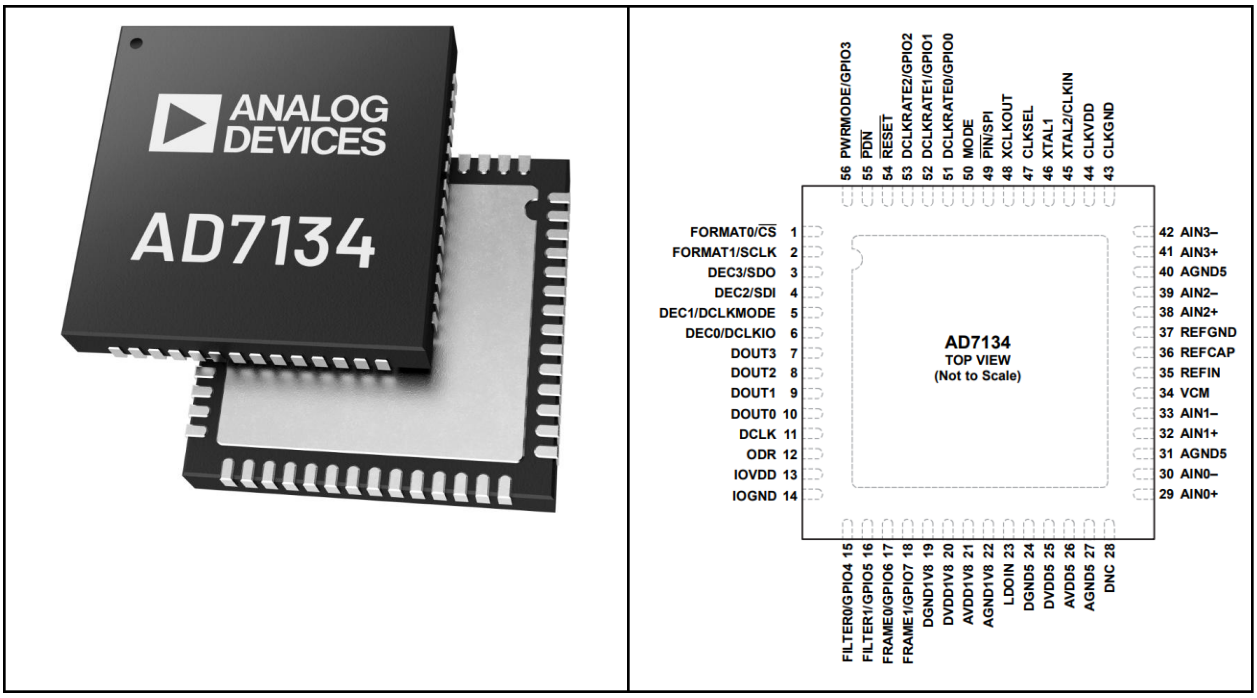

Figure 13. Image of AD7134BCPZ-RL7 and its pin configuration [6].

3.1.3 Digital to Analog Converter (DAC)

A DAC is a device that converts digital signals to analog signals. Our device utilizes a DAC to convert processed digital signals back into the analog domain. A DAC has two important parameters: setting time and resolution. Setting time is the time the DAC needs to receive the next value, which is related to the sampling rate and the resolution parameter is the same as the ADC. One thing to note is that the sampling rate of the DAC should be several multiples of the rate of the digital signal because of the Nyquist rate.

Proof-of-Concept

Similar to the ADC, we will also purchase commercial DACs and assemble them into our system. The DAC we have ordered is MCP4812, which has 2 input channels, 10-bit resolution, a sampling rate of 2.22 MHz, and can generate bipolar signals. The sampling rate is much greater than both ADCs, which avoids aliasing effects. It also supports SPI protocol so we can send digital signals from Raspberry Pi to this component.

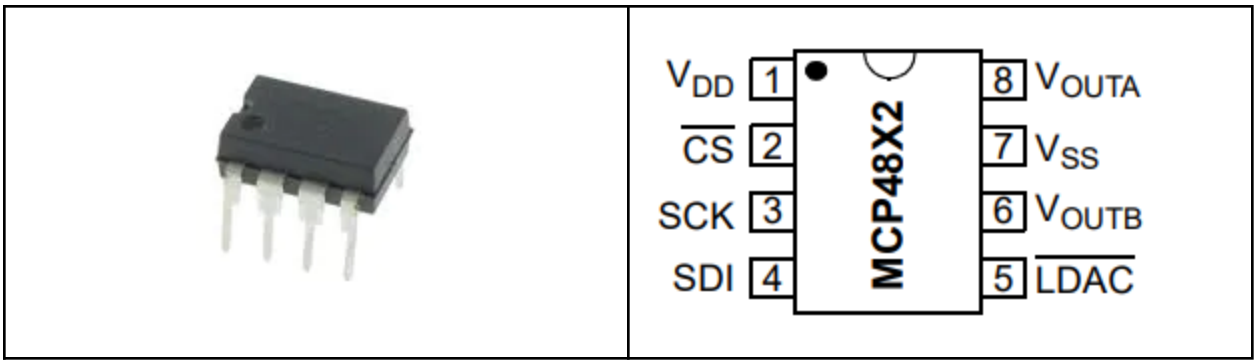

Figure 14. Image of MCP4812 and its pin configuration [7].

Below is a schematic showing how the MCP4812, and Raspberry Pi3 Model B will be wired for the proof of concept.

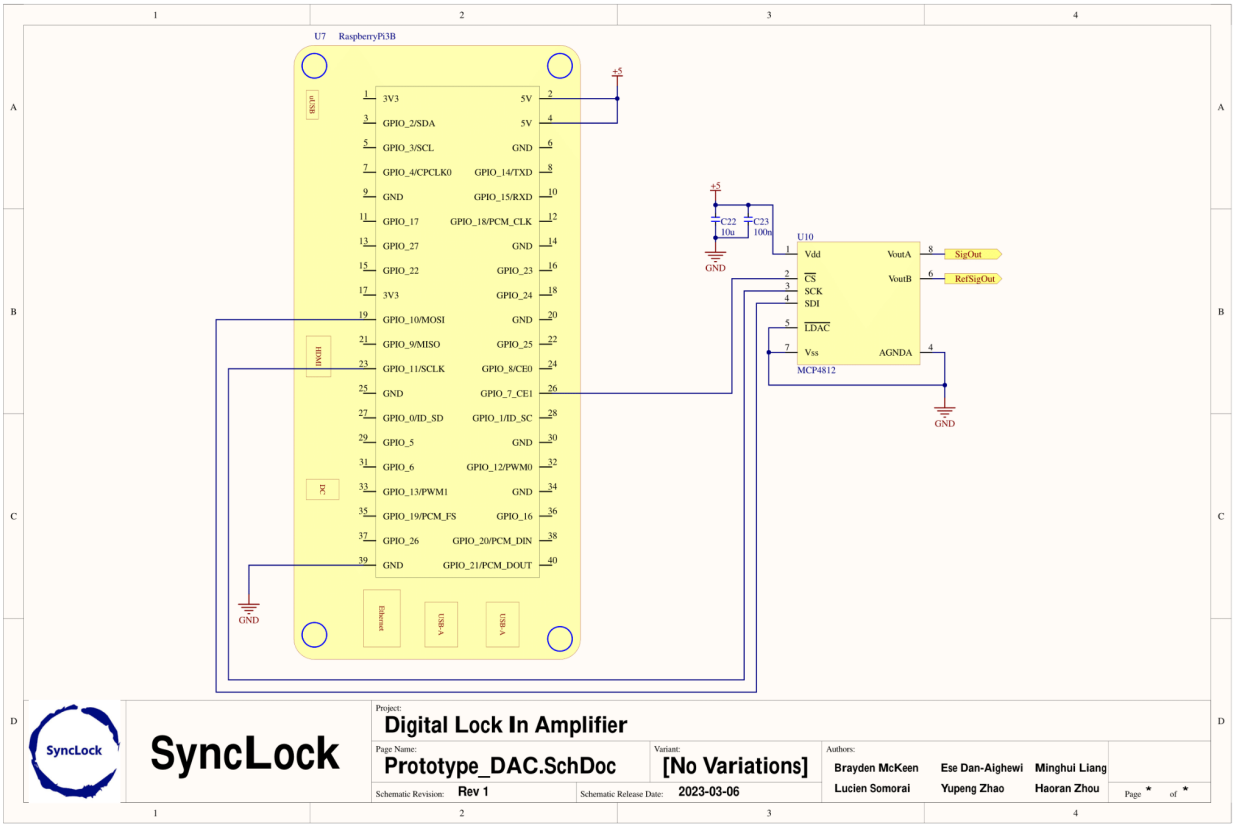

Figure 15. Schematic of MCP4812 and Raspberry Pi 3 Model B.

Prototype

In the prototype, we will switch the DAC to a higher resolution and change the sampling rate according to the sampling rates of ADCs used in the prototype design. The current choice is LTC1654CGN which has 2 input channels, 14-bit resolution, a sampling rate of 3.33 MHz, and is capable of generating bipolar signals. This selection is subject to change.

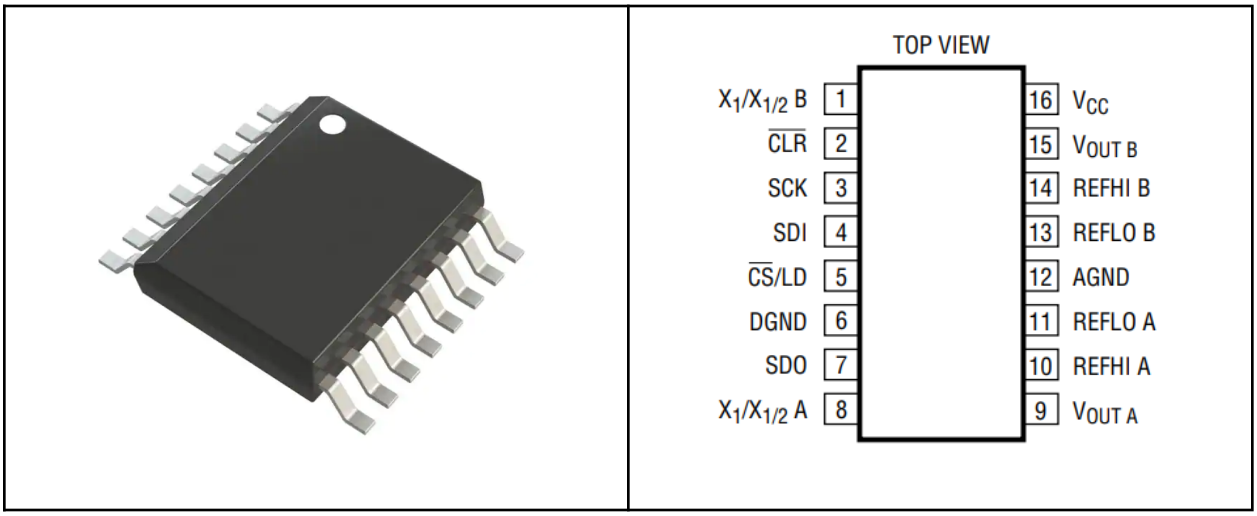

Figure 16. Image of LTC1654CGN and its pin configuration [8].

3.1.4 Processor

A processor is where the digital signals get processed. It mainly focuses on multiplication which is mixing and Fourier transform which is low-pass filtering. It is also responsible for sending control signals to each component.

Proof-of-Concept

In stage A, we simplified the importance of the processor by simply using Raspberry Pi, which has a Linux system installed on it. This drastically reduces the workload and time needed for the processor part. Figures 17 and 18 below show the Raspberry Pi 3 B+ model we are using and its pin configuration for sending and receiving digital signals.


Figure 17. Image of Raspberry Pi 3 B+ [9].

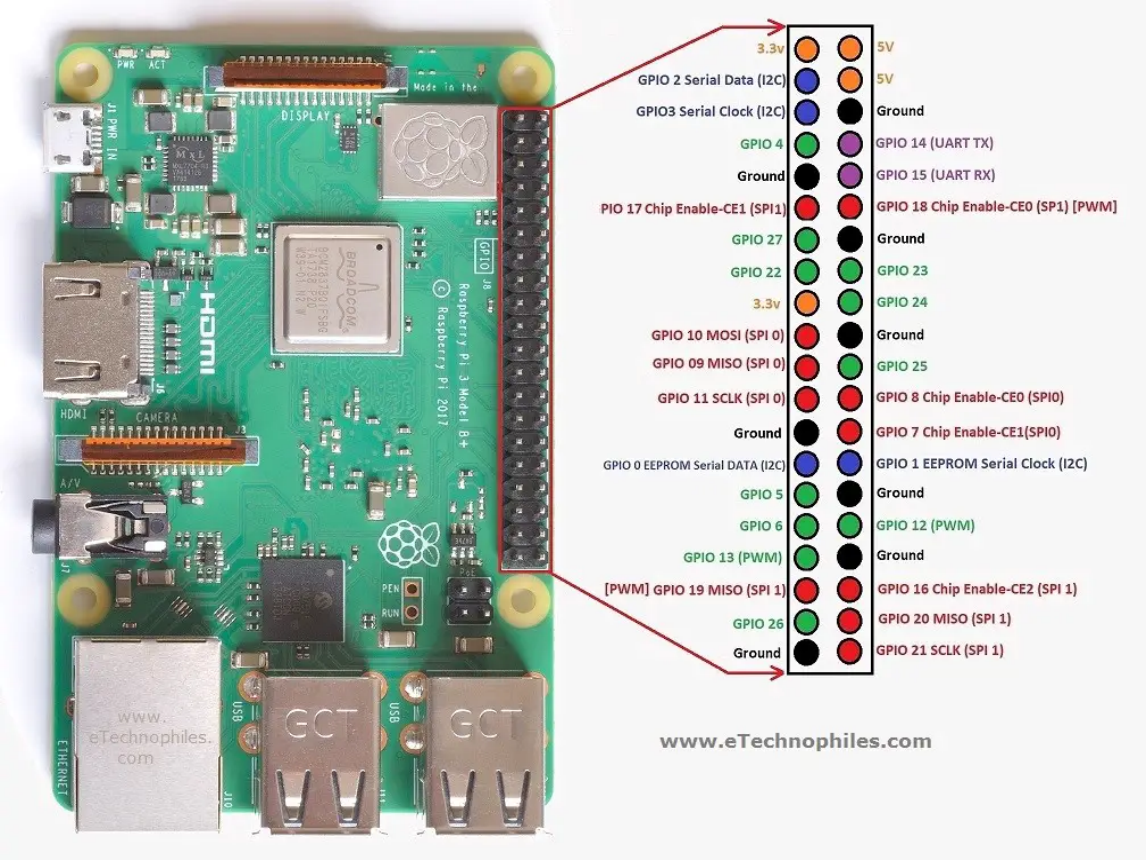

Figure 18. Pin configuration of Raspberry Pi 3 B+ [10].

Prototype

In stage B, we will switch from a Raspberry Pi to a microcontroller. This is because of one important requirement: R 2.2.3, the overall power consumption needs to be below 1 watt, whereas Raspberry Pi alone consumes more than that amount.

Currently, we haven't investigated the microcontroller models and their parameters.

3.1.5 Direct Digital Synthesizer (DDS)

A DDS is a component that generates analog and/or digital reference signals with a specific frequency. In common lock-in amplifiers, reference signals generated by a DDS are used to separate quadrature and in-phase components of the input signal.

Proof-of-Concept

Although a DDS is specified as a separate block in figure 2, in stage A, a DDS will be implemented through an internal Raspberry Pi process and a DAC. We will digitally generate the reference signal and mix it with the input signal. In this way, we can save time on hardware

development. More details of the DDS function are also discussed in the digital design specification section.

Prototype

For the final product, an external DDS chip will be used to generate the reference signal. The current choice is AD9837 which has 28-bit resolution and runs on a 16 MHz clock. It interfaces with the microprocessor via SPI and the power consumption is 8.5 mW at 2.3 V. Depending on the performance of the digital DDS function and microcontroller, we may choose to stick with the same design in stage A.

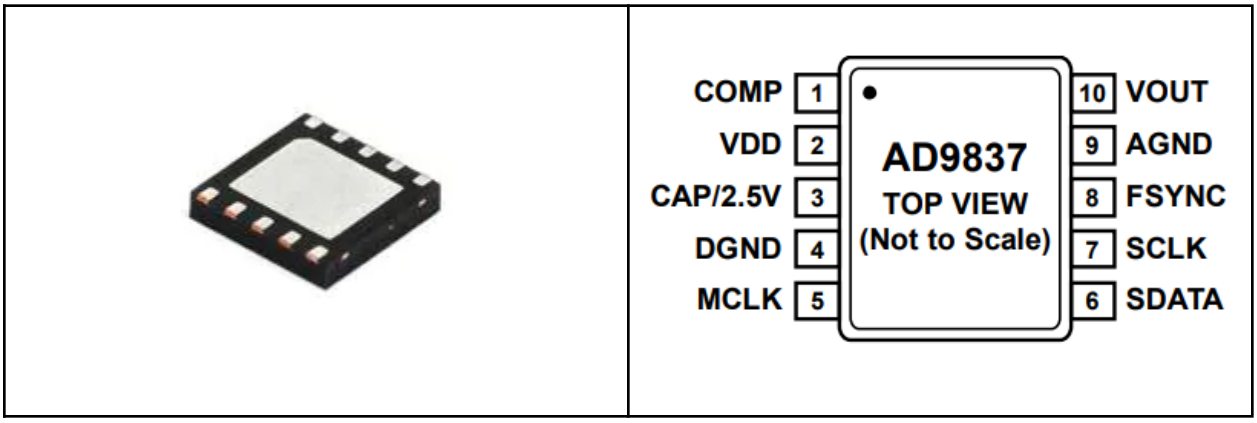

Figure 19. Image of AD9837 and its pin configuration [11].

Below is a schematic showing how the AD9837 DDS and Raspberry Pi3 Model B will be wired for the proof of concept.

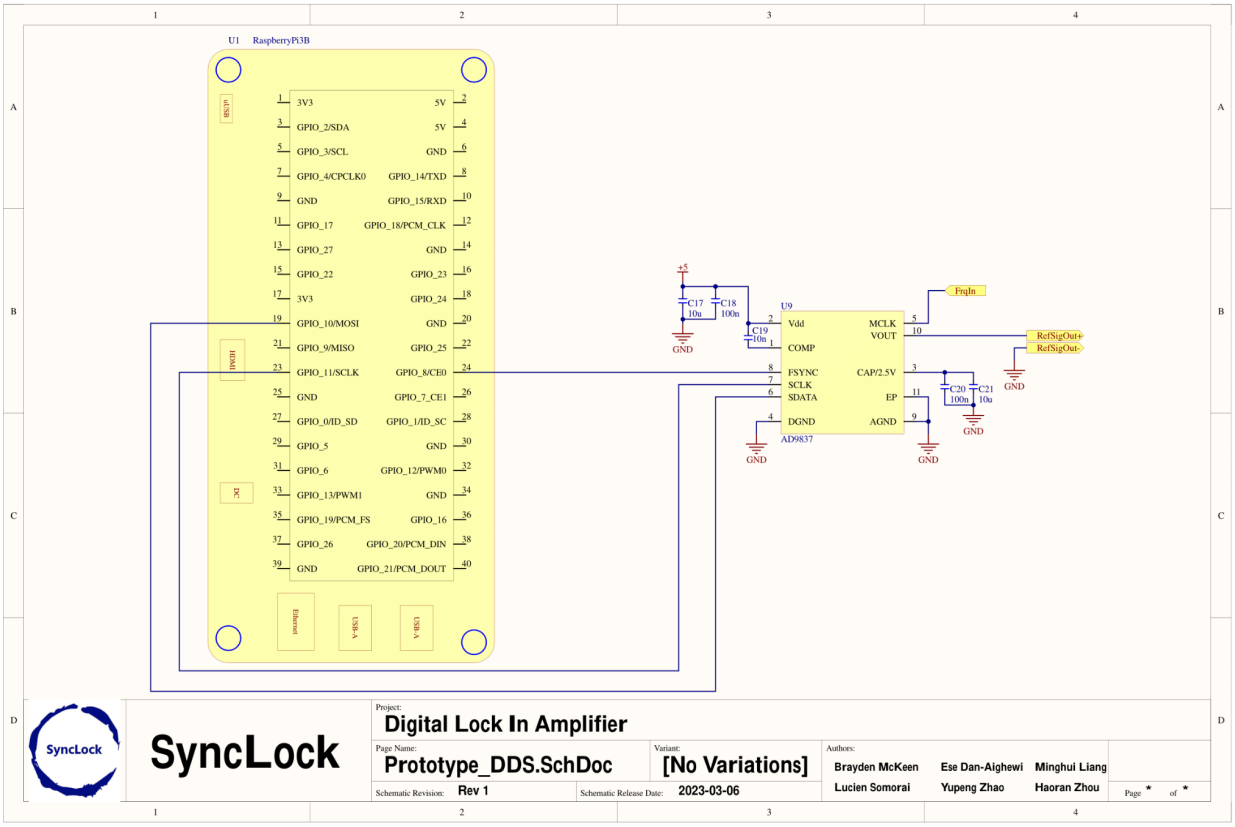

Figure 20. Schematic of AD9837 and Raspberry Pi 3 Model B.

3.1.6 Power Supply and SD card

The power supply needs to be able to provide sufficient power to all components. An SD card needs to record the data processed by the microprocessor.

Proof-of-Concept

In stage A, an integrated power supply or SD card will not be implemented. Since the Raspberry Pi is our microprocessor, it is powered by the USB-c connector and has an SD card. The other components are powered by a lab power supply.

Prototype

In stage B, all the parts, including a power supply and SD card, will be combined on one custom PCB. The power supply is powered by 3 rechargeable batteries. To implement an SD card, an off-the-shelf module will be mounted on the custom PCB.

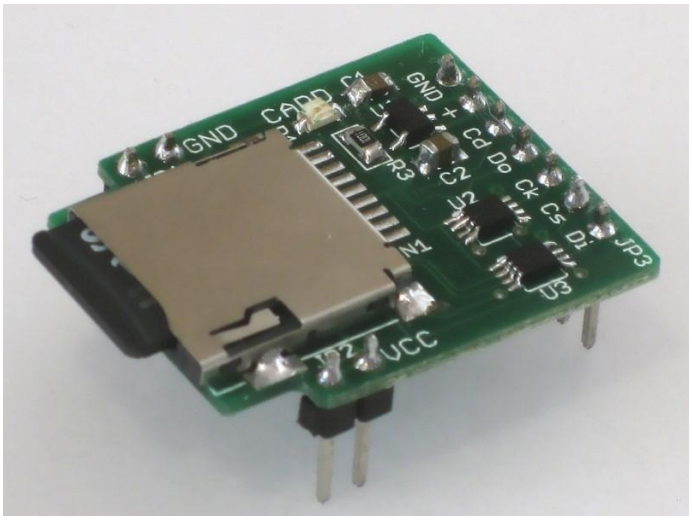

Figure 21. SD card PCB board [12].

3.2 Digital Design Specifications

The digital signal processing (DSP) unit at the proof-of-concept stage involves a direct digital synthesizer (DDS), mixers, and lowpass filters. They are implemented digitally in a Raspberry Pi board. Figure 22 shows the schematic of lock-in amplification. After the input signal is converted from the analog to the digital domain, it is received by the processor and mixed with the reference signal, which is generated by the DDS. The mixer outputs are low-pass filtered to block the noise and the 2ω frequency part. The top line produces the in-phase component X and the bottom line produces the quadrature component Y. The amplitude and the phase of the output are computed using a transformation from the Cartesian to the polar coordinate [13].

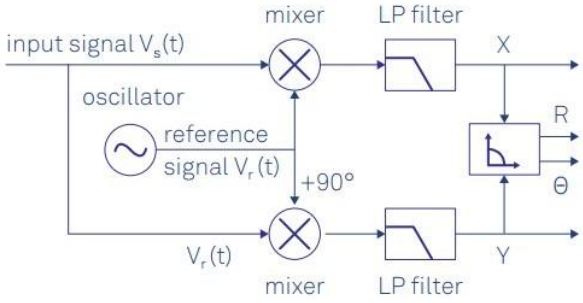

Figure 22. Schematic of lock-in amplification [13].

3.2.1 Direct Digital Synthesizer

The reference frequency generation is realized using a direct digital synthesizer (DDS). For the proof-of-concept stage, the DDS is coded as part of the digital processing unit in the microprocessor. As shown in figure 23, the DDS consists of two major parts, including a phase accumulator and a look-up table.

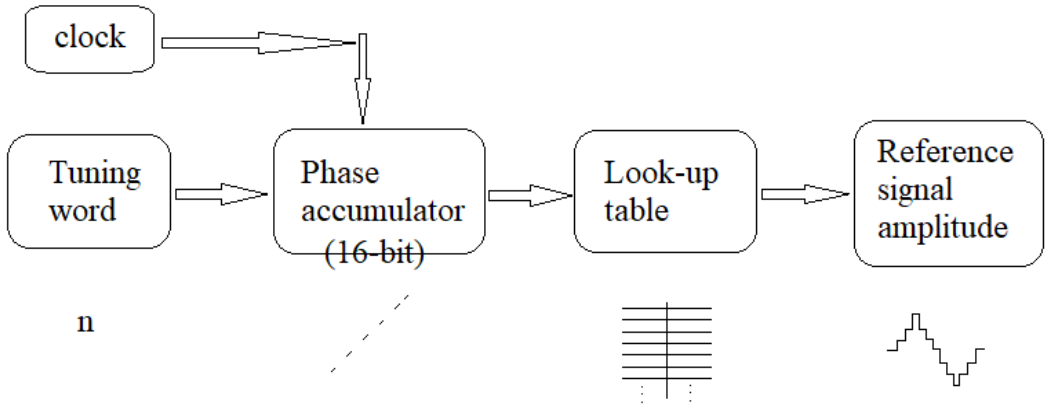

Figure 23. The block diagram of a DDS.

The DDS system runs on a clock that causes the phase accumulator to count up. The clock is provided by the microprocessor. The 16-bit phase accumulator stores 2^{16} indices, and the look-up table corresponds these indices to 2^{16} distinct values over a sine wave period on a one-to-one basis. As the phase accumulator increases, indices are sent to the look-up table, and the mapped sine wave value is sent to the output. Cycling through the indices from 0 to ($2^{16} - 1$) completes one reference signal period.

The tuning word controls the frequency of the reference signal generated by the DDS. While the tuning word is set to 1, the phase accumulator increments the index by 1 without skipping. All sine wave values stored in the look-up table are sent. If the tuning word is set to some other positive integer value n, the phase accumulator would count by n and reach the end of the cycle at a faster rate. Figure 24 shows the code snippet of the DDS simulation in Python. The simulation duration is set to 1 second. Figure 24 bottom left has the tuning word set to 2 and the output is the reference signal at a frequency equals to 2 Hz. The bottom right shows the tuning word set to 5 corresponds to a frequency at 5 Hz. While the sampling frequency equals the look-up table length, the DDS is capable of generating reference signals with a maximum frequency at half of that. In this simulation, the sampling frequency is restricted by the array size but when writing to the microprocessor, it will match the clock rate of the microprocessor.

```python
# Direct Digital Synthesizer (DDS)
#
# The DDS generates reference signal at the frequency set by the
#    tuning word. By incrementing the index, the corresponding
#    digital value in the look-up table is sent to the output.

import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline

# Sine look-up table (LUT)
# Stores sine wave value for one period for duration of one second
TABLE_LEN = 2**16
indN = np.arange(0, TABLE_LEN, 1)
sinLUT = np.sin(2*np.pi*indN/TABLE_LEN)

DURATION = 1 # simulation length in seconds
SAMPLING_F = TABLE_LEN

tuningWord = 2
i = 0 # counter index
j = 0
outputTime = np.zeros(SAMPLING_F*DURATION)
outputSine = np.zeros(SAMPLING_F*DURATION)
while True:
    outputTime[j] = j/SAMPLING_F
    outputSine[j] = sinLUT[i%TABLE_LEN]
    if j >= SAMPLING_F*DURATION-1:
        break
    i += tuningWord
    j += 1

plt.plot(outputTime, outputSine)
```

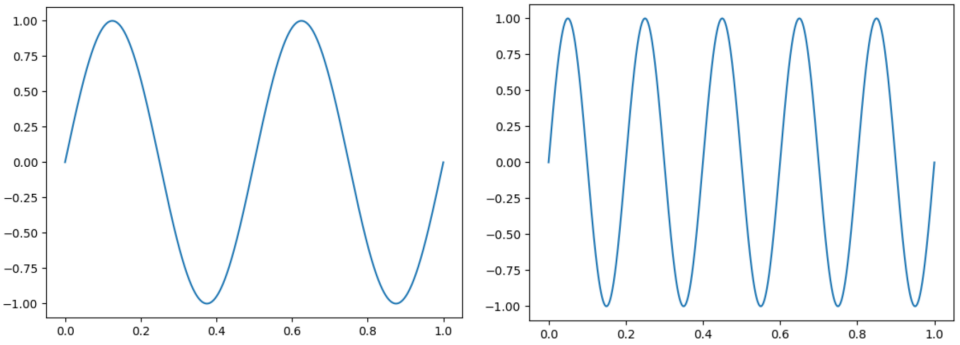

Figure 24. DDS simulation in Python. Code snippet (top). Reference signal at 2 Hz (bottom left). Reference signal at 5 Hz (bottom right).

16 bits are allocated for the phase accumulator to match the resolution of the ADC used for the proof-of-concept stage. For the final product, the team would like to connect an external DDS chip to generate the reference signal. The current choice is AD9837 which has a 28-bit resolution and runs on a 16 MHz clock. It interfaces the microprocessor via SPI and the power consumption is 8.5 mW at 2.3 V.

3.2.2 Mixer

The mixer operates by multiplying the input signal with the reference signal. Assuming the input signal has frequency ω and amplitude A, that $v_{in} = A \cdot \sin(\omega t)$. The reference signal will also have frequency ω and can be written as $v_{ref} = \sin(\omega t)$. Mixing the two signals will yield

$$v_{mix} = A \cdot \sin(\omega t) \cdot \sin(\omega t) = A/2 \cdot (1 + 2\cos(2\omega t)),$$

Equation 2

where the following identity is inserted

$$\sin^2(\theta) = (1/2) \cdot (1 - \cos(2\theta)).$$

Equation 3

Low Pass filtering the mixed signal will remove the AC component and keep the DC component, which is A/2. Scaling the DC value by 2 will restore the amplitude carried by the input signal.

Now, suppose the input signal has a noise component at a different frequency ω'. Mixing it with the reference gives

$$\sin(\omega' t) \cdot \sin(\omega t) = (1/2) \cdot [(\cos(\omega' - \omega)t) + (\cos(\omega' + \omega)t)].$$

Equation 4

The high-frequency component, again, will be blocked by the low pass filter. Only noise signal with a fixed frequency that is close to the reference frequency ω will add to the DC component.

3.2.3 Low Pass Filter

A lowpass filter allows low-frequency signals to pass through while blocking the high-frequency signals. It imposes a cutoff frequency where signals with frequencies higher than that will have amplitude reduced by at least 3 decibels.

In the digital lock-in amplifier, the algorithm used for the lowpass filter is the finite impulse response (FIR) filter. It uses a finite number of coefficients to compute the output signal and provides several advantages including linear phase response, a low ripple in the pass band, and good control over the filter characteristics such as the transition bandwidth.

The design of the FIR filter utilizes the following two theorems:

1. The Fourier Transform of the impulse response of a system is the frequency response of this system.

2. The convolution of two functions in the time domain is equivalent to the multiplication of the two functions in the frequency domain.

From theorem 1, the design can start with a desired frequency response that outlines the filter's passband, the transition band, and the stopband characteristics, where the impulse response of the filter is obtained by performing an inverse Fourier Transform. Figure 25 left shows a square pulse function that has a value 1 around the origin, from DC to the low cutoff frequency and 0 everywhere else. The square function works as an ideal lowpass filter as it shows no ripple in the passband and the stopband, and the transition band has zero bandwidth. Its inverse Fourier Transform is shown in the middle, which is a sinc function centered at zero.

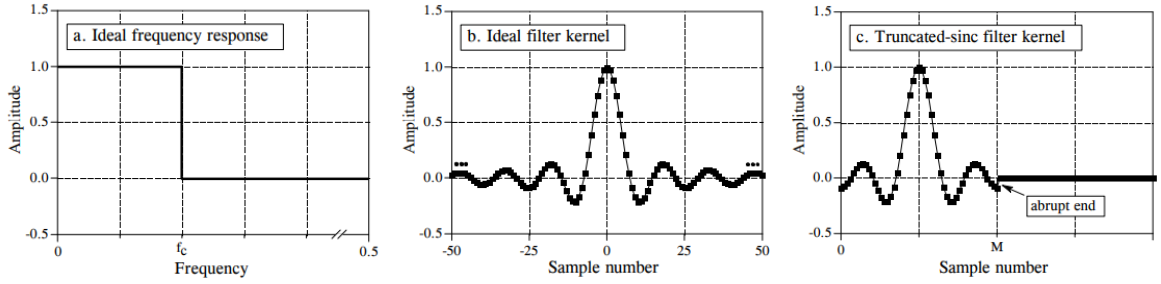

Figure 25. Ideal LPF with square function in the frequency domain (left). Sinc function in the time domain (middle). Shifted and chopped impulse response in discrete time (right) [14].

Issues with using the sinc function as the impulse response of the lowpass filter are listed as the following:

- The system is non-causal: The impulse response function has non-zero values below time equals zero.
- Infinite length for convolution: The function has a non-zero value at infinite, which makes computation unrealistic.
- Continuous time function: The function is in the analog domain, where digital signal processing requires discrete functions.

To address the above issues, the system needs to shift to the right such that all non-zero values are in the positive time domain. The tails of the Sinc function need to be stretched flat. Also, the function needs to be discretized by applying appropriate sampling. The resulting impulse response is shown in figure 25 right.

This brings up other issues in the filter's frequency response including ripples in the passband and side lobes passing the transition band. A window function would be mixed with the impulse response to smooth the ripples. Figure 26 left shows the Blackman window in the discrete time domain. It exhibits a bell-shaped curve from 0 up to sample M, and stays zero-valued above sample M. By multiplying the Blackman window with the shifted and chopped sinc, the filter will have a smooth passband characteristic and it is shown in figure 26 middle and right [14].

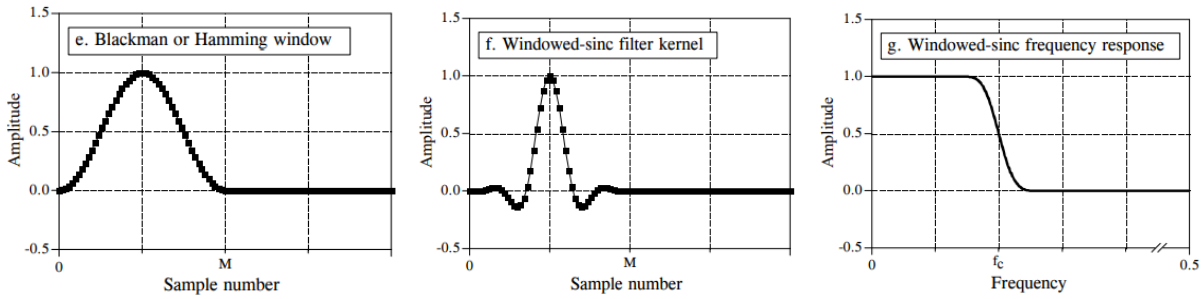

Figure 26. Blackman window in discrete time (left). Windowed-sinc impulse response (middle). Windowed-sinc frequency response (right) [14].

According to theorem 2, multiplication in the frequency domain is equivalent to convolution in the time domain. Hence, to implement the filtering action, the discrete-time convolution is applied between the input signal and the lowpass filter. While the filter has length M and the input signal is stored in a buffer array x, the output signal can be computed as the following.

$$y[n] = \sum_{i=0}^{M-1} h[i] \cdot x[n-i]$$

Equation 5

The input signal buffer is implemented using the circular buffer technique to improve the read/write efficiency during signal processing [15]. The following lists a summary of the lowpass filtering implementation:

- Store the latest sample in the input buffer.
- Increment the input buffer index.
    - If the buffer index exceeds the filter length, reset the index to 0.
- Compute the output using convolution.
    - Multiply the input sample by the filter sample.
    - Accumulate the result to the output.
    - Shift the input sample index and the filter sample index.
- Return the filtered output sample.

The prototype development will involve replacing the Raspberry Pi board with a microprocessor chip that consumes less power. The DSP algorithm will remain the same and the code implementation will switch accordingly.

3.3 Software Design Specifications

The software design includes specifications of how the DLA application is set up and used with the Raspberry Pi, as well as the interface from the Raspberry Pi to the ADC using python packages and libraries. The design requirements are specified in the table below:

ID	Tag	Requirement Description	Changes For Future Designs	Req ID
Des 3.3.1	A	Software will be able to adjust the amplitude, frequency, and phase of reference signal	None	Req 2.5.1A
Des 3.3.2	A	Software will be able to configure amplifier parameters	None	Req 2.5.2A
Des 3.3.3	A	Software will be able to read the data from SD card or the device directly	None	Req 2.5.3A
Des 3.3.4	A	Software will include a help menu to allow a user access directions on how to use the DLA and application	None	Req 4.8 Req 4.9
Des 3.3.5	B	Software should plot frequency spectrum of chosen signal	None	Reg 2.5.4B
Des 3.3.6	B	Software will be able to report faults from the PCB	None	Req 2.5.5B
Des 3.3.7	C	User will be able to save data for the further applications	None	Req 2.5.6C

Table 5. Software design specifications and related requirements.

3.3.1 User Interface Design

To set up the DLA device for use, the user will input configuration values to create a reference signal that will be mixed with the input signal. An interface, in the form of an application, that allows for this will be loaded once the microcontroller is plugged into a computer. A use case diagram below shows the relationship between a user and the application, as well as the functions and scope of the system.

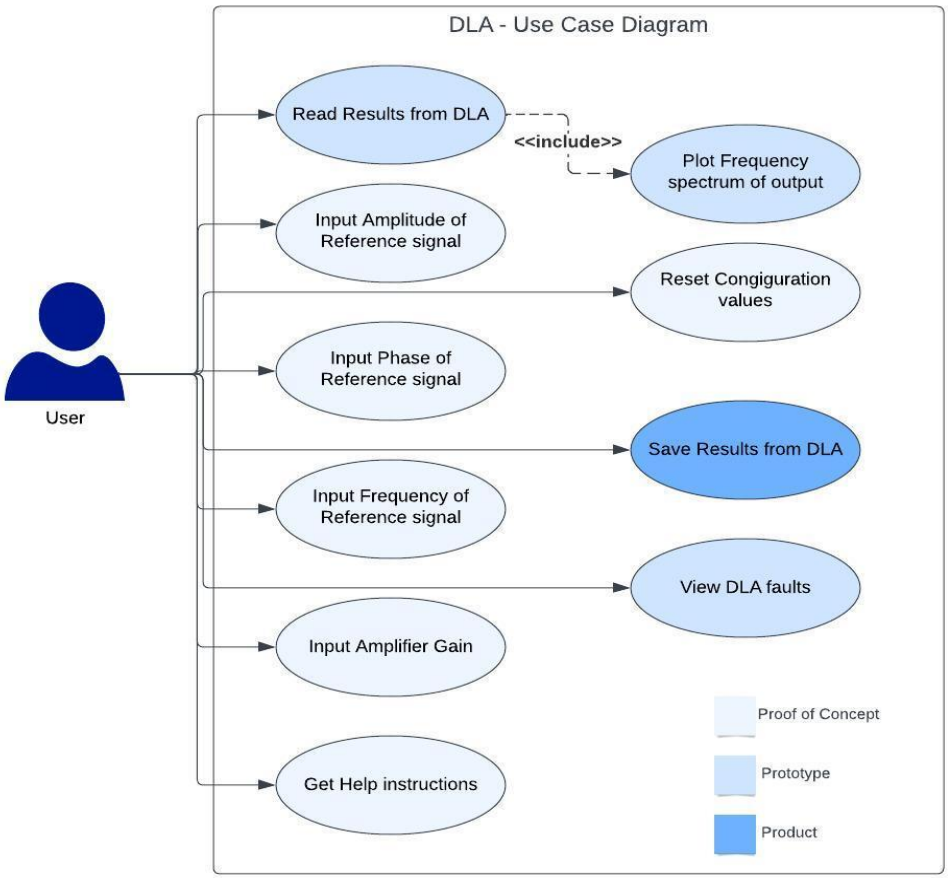

Figure 27. Use Case Diagram showing high-level functions of the application.

Loading up the application will bring up the main screen, where the user will be prompted to input the amplitude, phase, and frequency of the desired reference signal. There will be an error notification if a value outside the range of each value is entered. The pre-amplifier will also require an input for the gain, to boost the input signal. An option to reset all the input values will be at the bottom of the screen to allow the user to start afresh. All input values will be cleared, and preset values will replace them. These preset values will also replace any wrong input while the user is changing the configuration values and will also be loaded on startup to always allow for normal operation.

Once the Raspberry Pi can communicate with all the components on the breadboard and produce digital output, a frequency spectrum graph will be displayed on the screen as an output of the DLA. All the components described above will be programmed in the proof-of-concept stage, including a feature that provides help instructions.

For the prototype, the Raspberry Pi will be replaced with a microcontroller and will be connected to a DAC, which will produce analog values as an output. These values will be saved on an SD card and the application will allow the user to read the results from there, and possibly perform even more analysis on the results. There will also be an option to read errors from the device if it does not function properly

3.3.2 Application Development

For the proof-of-concept, MATLAB will be used to develop an application for the DLA. It is compatible with Raspberry Pi, and it also provides a great platform to represent the output from the DLA. Creating a standalone MATLAB application does not require a license or MATLAB to be installed on the user's computer, however, it does require the installation of MATLAB Compiler Runtime (MCR) [16] in the same version as that of the compiler used to develop the application. Using MATLAB will allow for other signal processing capabilities that can be used to design the prototype stage.

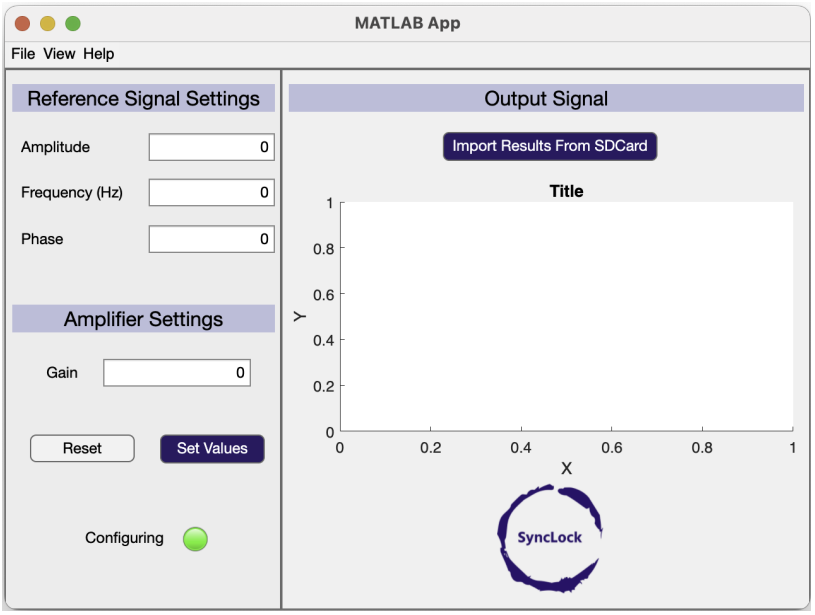

Figure 28. UI design of the DLA application.

The window in figure 27 above shows a capture of the application using MATLAB's App Designer. It creates a '.mlapp' file which formats all the components on the screen [17] and allows callbacks to be added to program the behavior of the components. The following image shows an example of the callback function used to initialize the amplitude input field and attach its value to a global variable called 'app.Amplitude', which will be stored and used for configuring the DLA. The updataAppLayout function allows for flexibility on the screen when the graph layout has to be updated.

```
    % Callbacks that handle component events
    methods (Access = private)

        % Value changed function: AmplitudeEditField
        function AmplitudeEditFieldValueChanged(app, event)
            value = app.AmplitudeEditField.Value;
            app.Amplitude = value;
        end

        % Changes arrangement of the app based on UIFigure width
        function updateAppLayout(app, event)
            currentFigureWidth = app.UIFigure.Position(3);
            if(currentFigureWidth <= app.onePanelWidth)
                % Change to a 2x1 grid
                app.GridLayout.RowHeight = {429, 429};
                app.GridLayout.ColumnWidth = {'1x'};
                app.RightPanel.Layout.Row = 2;
                app.RightPanel.Layout.Column = 1;
```

Figure 29. Code snippet from App Designer, initializing callback functions.

The application file can be packaged with MATLAB Compiler, which will create an executable '.exe' file. Although it can't be run on Raspberry Pi, it will communicate with it through a configuration file that the Raspberry Pi can read.

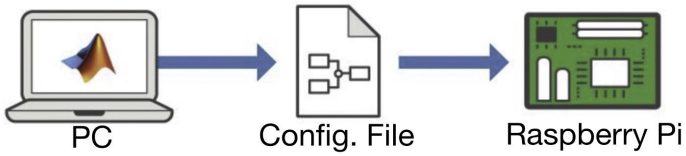

Figure 30. Interface between Raspberry Pi and DLA application. Adapted from [18]

For the prototype stage, MATLABs Hardware Manager can be used to communicate with the microcontroller. Support packages and add-ons downloaded through the Serial Explorer will allow communication through the serial ports of the microcontroller. MATLAB's Coder app can be used to generate C/C++ code, which can make up for the limitations of the App Designer.

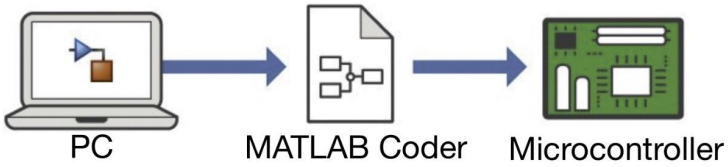

Figure 31. Interface between Microcontroller and DLA application. Adapted from [18]

3.3.3 Raspberry Pi and ADC

The Raspberry Pi must be updated to the latest version, should run Raspbian, and be able to connect to the terminal via SSH [19]. The ADC will communicate with it through the SPI serial connection, and use its GPIO SPI pins connected with software SPI. The ADC is wired to the Raspberry Pi as follows:

ADC	Raspberry Pi
VDD	3.3V
VREF	3.3V
AGND	GND
DGND	GND
CLK	Pin 18
DOUT	Pin 23
DIN	Pin 24
CS/SHDN	Pin 25

Table 6. Wire configuration between Raspberry Pi and ADC.

A python version and python package index, including the corresponding ADC library, are installed on the Raspberry Pi, and the ports are set up in a python file.

```
1  # Configuring software SPI for ADC
2  CLK = 18
3  MISO = 23
4  MOSI = 24
5  CS = 25
6  mcp = Adafruit_MCP3008.MCP3008(clk=CLK, cs=CS, miso=MISO, mosi=MOSI)
```

Figure 32. Setting up the ADC in Raspberry Pi terminal. Adapted from [19]

Running the code above sets up the ports and prints out the values from the ADC channels after every 0.5 seconds [19]. Using "**read_adc(channel)**" from the index library will take in a channel and read whatever output it has at that time.

For the prototype stage, the Raspberry file will be replaced with a microcontroller and the code will have to be changed to support all the components on the board, including the DAC, DDS, SD Card, and USB controller.

3.4 Mechanical Design Specifications

ID	Tag	Description	Changes For Future Designs	Req ID
Des 3.4.1	C	The device will be portable	Relocate components to allow for a thinner enclosure	R 2.2.5 R 2.2.6
Des 3.4.2	C	The enclosure will provide protection for the internal PCB and batteries	Change lid screw positions to avoid sharp bends in O-ring	R 2.2.8 R 2.2.9 R 2.2.10 R 2.2.11
Des 3.4.3	C	The device will require minimal tools or equipment to operate	None	R 2.2.7 R 2.2.12

Table 7. Mechanical design specifications and related requirements.

3.4.1 Mechanical Enclosure Design

A 3D printable enclosure was designed to facilitate the field use of the device. To be used in the field, the enclosure must provide the user with three things: portability, protection, and ease of use. To address portability, the device, when mounted in its enclosure, has overall dimensions of 15 cm long, 8cm wide, and 13 cm thick. The figure below demonstrates the device mounted and sealed in its enclosure.

Figure 33. Field Ready Device Enclosure.

Rubber corner bumpers will provide impact protection and also resist sliding when placed on non-level surfaces. O-ring gaskets between the main housing and the battery panels and lid prevent water or dust ingress and all electrical components on the exterior of the enclosure are rated at IP67 or higher.

For ease of use, all screws are M2x0.4 with 1.5mm sockets heads, meaning the user need only have one tool to be able to access both the enclosed PCB and batteries. The device is also powered off of commonly available D-cell batteries. The figure below provides an exploded view of the enclosure showing how the screws, gaskets, PCB, and enclosure pieces come together.

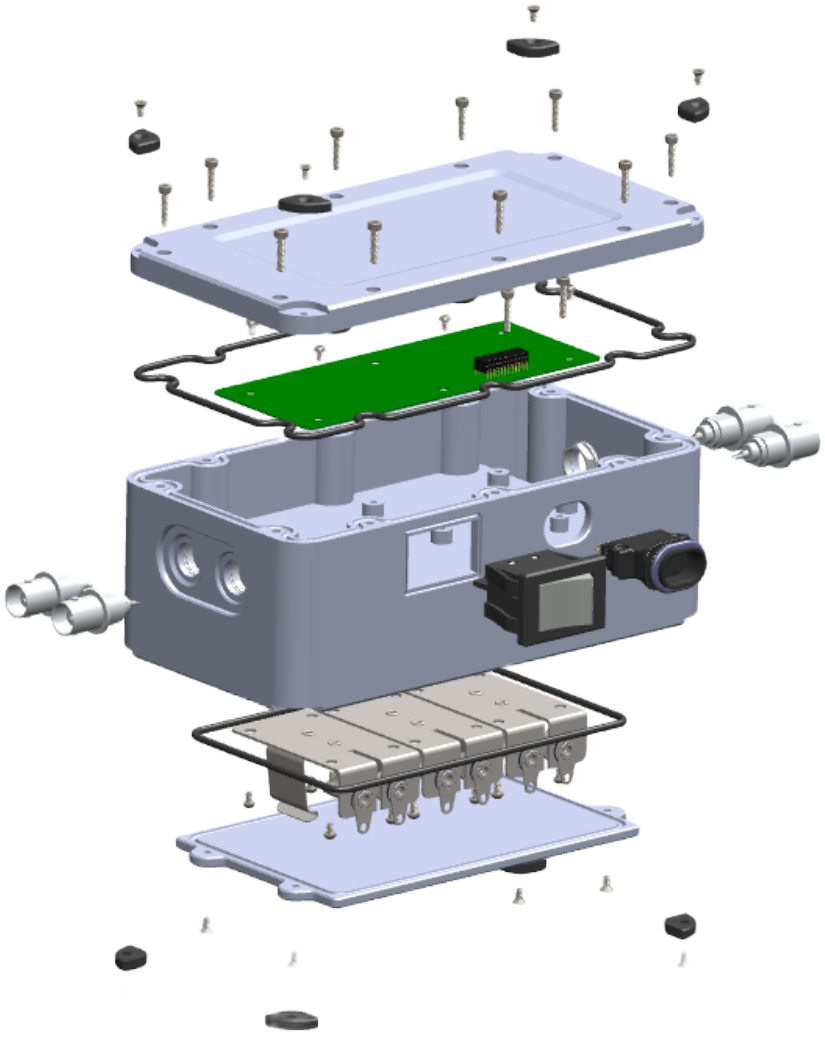

Figure 34. Exploded View of Device Enclosure.

3.4.2 PCB Design

The overall dimensions of the PCB are 106mm long, by 46mm wide. The figure below shows how the PCB is mounted in the enclosure. To mount the PCB in the enclosure, it will be dropped in with the lid removed, and slid into place connecting the onboard connector with the enclosure-mounted USB-C connector. Finally, screws are installed to secure the PCB in its place.

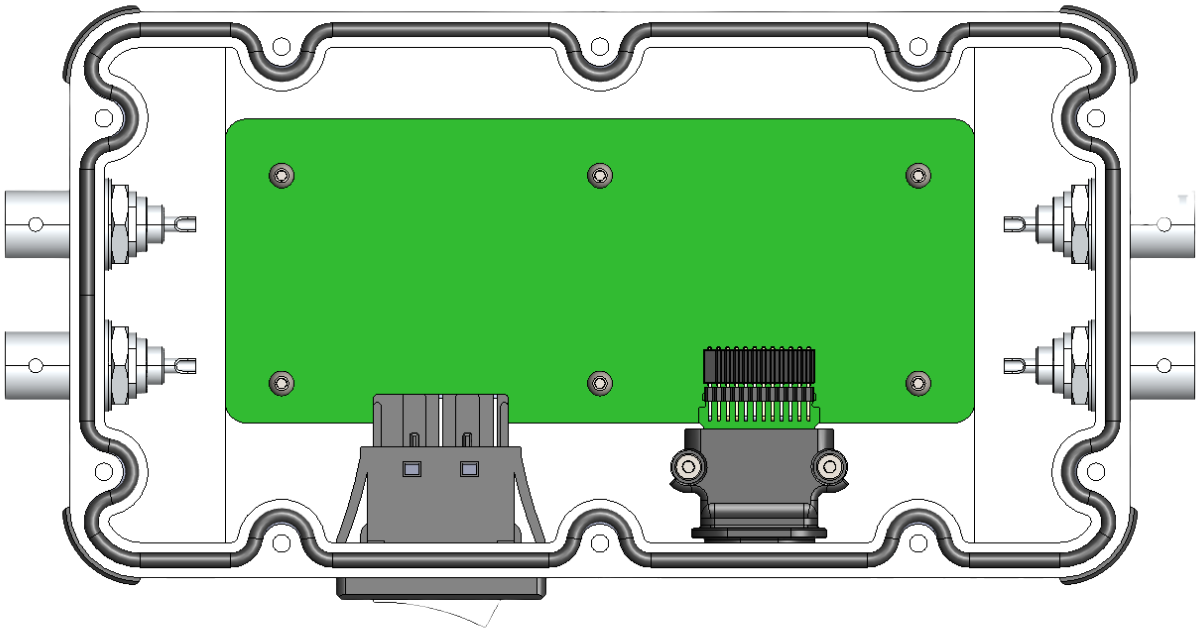

Figure 35. View of Enclosure Interior.

4 Conclusion

This document outlines the specifications and requirements for each aspect of the design of a digital lock-in amplifier. Specific components included hardware, digital, software, and mechanical requirements which are summarized below.

1. Hardware Requirements
    a. Pre-Amp: amplifies the input signal to an appropriate size for the ADC
    b. Analog to Digital Converter: converts analog signals to digital ones
    c. Digital to Analog Converter: converts digital signals to analog ones
    d. Processor: Digital signal processing
    e. Direct Digital Synthesizer: generates reference signals with a specific frequency
    f. Power Supply and SD card: provides power to the components and stores data

2. Digital Requirements
    a. Direct Digital Synthesizer: generates reference signals with a specific frequency
    b. Mixer: multiplies input signal with reference signal
    c. Low Pass Filter: filters out high frequencies

3. Software Requirements
    a. User Interface Design: allows user to input configuration values creating a reference signal
    b. Application Development: Matlab application
    c. Raspberry Pi and ADC: serves as the microcontroller for the prototype stage

4. Mechanical Requirements
    a. Mechanical Enclosure Design: 3D printable enclosure providing portability, protection, and ease of use
    b. PCB Design: A PCB that fits within the enclosure and is easily mounted or dismounted

In addition, this document will also be a guideline through the proof-of-concept stage, prototyping stage, and production stages. While we work through the design we might modify the design and select different components based on our development. The Synclock company will dedicate more efforts toward the design and continue to develop a product that is functional, intuitive, and safe for all users.

5 References

[1] Texas Instruments, “INA821 35-µV Offset, 7-nV/√Hz Noise, Low-Power, Precision Instrumentation Amplifier,” SBOS893D datasheet, Aug. 2018 [Revised Jun. 2020]. Available: https://www.ti.com/lit/ds/symlink/ina821.pdf. [Accessed: Mar. 17, 2023].

[2] Texas Instruments, “INA823 Precision, Low-Power, Wide-Supply (2.7-V to 36-V) Instrumentation Amplifier,” SBOSA75B datasheet, Jul. 2021 [Revised Nov. 2002]. Available: https://www.ti.com/lit/ds/sbosa75b/sbosa75b.pdf?ts=1679111243211. [Accessed: Mar. 17, 2023].

[3] Texas Instruments, “INA819 35-µV Offset, 8-nV/√Hz Noise, Low-Power, Precision Instrumentation Amplifier,” SBOS959D datasheet, Dec. 2018 [Revised Apr. 2022]. Available: https://www.ti.com/lit/ds/symlink/ina819.pdf?HQS=dis-dk-null-digikeymode-dsf-pf-null-wwe&ts=1677917013869&ref_url=https%253A%252F%252Fwww.ti.com%252Fgeneral%252Fdocs%252Fsuppproductinfo.tsp%253FdistId%253D10%2526gotoUrl%253Dhttps%253A%252F%252Fwww.ti.com%252Flit%252Fgpn%252Fina819 . [Accessed: Mar. 17, 2023].

[4] “MCP3008 | Microchip Technology.” [Online]. Available: https://www.microchip.com/en-us/product/MCP3008. [Accessed: 17-Mar-2023].

[5] “ADS1115,” ADS1115 data sheet, product information and support | TI.com. [Online]. Available: https://www.ti.com/product/ADS1115?keyMatch=ADS1115&tisearch=search-everything&usecase=GPN. [Accessed: 17-Mar-2023].

[6] “Mouser Electronics - AD7134,” Mouser Electronics. [Online]. Available: https://www.mouser.ca/ProductDetail/Analog-Devices/AD7134BCPZ-RL7?qs=P1JMDcb91o6OEjrJrhp8lg%3D%3D. [Accessed: 17-Mar-2023].

[7] “Mouser Electronics - MCP4812,” Mouser Electronics. [Online]. Available: https://www.mouser.ca/ProductDetail/Microchip-Technology-Atmel/MCP4812-E-P?qs=bxUt0k7cytI17caZuUAwlA%3D%3D. [Accessed: 17-Mar-2023].

[8] “LTC1654CGN#PBF: Digi-key electronics,” Digi. [Online]. Available: https://www.digikey.ca/en/products/detail/analog-devices-inc/LTC1654CGN-PBF/891204. [Accessed: 17-Mar-2023].

[9] Raspberry Pi, “Buy A Raspberry pi 3 model B+,” Raspberry Pi. [Online]. Available: https://www.raspberrypi.com/products/raspberry-pi-3-model-b-plus/. [Accessed: 17-Mar-2023].

[10] eTechnophiles. "Raspberry Pi 3 B+ Pinout with GPIO Functions, Schematic, and Specs in Detail." Available online: https://www.etechnophiles.com/raspberry-pi-3-b-pinout-with-gpio-functions-schematic-and-specs-in-detail/ (accessed on March 17, 2023).

[11] “AD9837BCPZ-RL7: Digi-Key Electronics,” *Digi*. [Online]. Available: https://www.digikey.ca/en/products/detail/analog-devices-inc/AD9837BCPZ-RL7/2700194. [Accessed: 17-Mar-2023].

[12] P. byDP, “PCB a week 19: SD Card Breakout for Breadboards,” Dangerous Prototypes, 01-Aug-2013. [Online]. Available: http://dangerousprototypes.com/blog/2013/08/01/pcb-a-week-19-sd-card-breakout-for-breadboards/. [Accessed: March 17, 2023].

[13] “Principles of Lock-in Detection,” *Zurich Instruments*, Nov. 2016. [Online]. Available: https://www.zhinst.com/americas/en/resources/principles-of-lock-in-detection. [Accessed: Mar. 16, 2023].

[14] S. W. Smith, “Windowed-Sinc Filters,” in *The Scientist and Engineering's Guide to Digital Signal Processing*, 2nd ed. California Technical Publishing, 1999.

[15] S. Arar, “Circular Buffer: A Critical Element of Digital Signal Processors,” *All About Circuits*, Nov. 13, 2017. [Online]. Available: https://www.allaboutcircuits.com/technical-articles/circular-buffer-a-critical-element-of-digital-signal-processors/. [Accessed: Mar. 14, 2023].

[16] “Getting started: Standalone applications using MATLAB Compiler - Video,” *Video - MATLAB*. [Online]. Available: https://www.mathworks.com/videos/getting-started-standalone-applications-using-matlab-compiler-100088.html. [Accessed: 15-Mar-2023].

[17] “Matlab app designer: Create desktop and web apps in MATLAB,” *MATLAB & Simulink*. [Online]. Available: https://www.mathworks.com/products/matlab/app-designer.html. [Accessed: 15-Mar-2023].

[18] “Raspberry pi programming with MATLAB and simulink,” *MATLAB & Simulink*. [Online]. Available:

https://www.mathworks.com/discovery/raspberry-pi-programming-matlab-simulink.html. [Accessed: 15-Mar-2023].

[19] T. Dicola, “Raspberry Pi analog to digital converters,” *DigiKey*. [Online]. Available: https://www.digikey.ca/en/maker/projects/raspberry-pi-analog-to-digital-converters/72388f5f1a0843418130f56c53a1276c. [Accessed: 15-Mar-2023].

6 Appendix: Design Alternatives

In this appendix, we will discuss some backup plans for our design.

Choice of Processor

Currently, we use Raspberry Pi as the processor in stage A and plan to use a microcontroller in stage B. In the last progress meeting, Dr. Michael suggested a portable FPGA board called step FPGA, it can also be a choice of the processor. However, I can’t find any power consumption information related to this FPGA. Based on the power requirement: Req 2.2.3, we are uncertain that the step FPGA can be a choice of processor.

Choice of ADC and DAC

Even though we introduced the ADC and DAC ordered and will order in the hardware specification, we actually created two lists of ADCs and DACs. According to the ADC and DAC requirements: from R 2.3.1 to R 2.3.11 and further updates from the clients, we will make different ADC and DAC choices in the prototype design.

Choice of Casing

Initially we thought about 3D printing a case for our device in stage B or C. Dr. Hegedus suggested buying a suitable IP 6/7 proof case and gaskets for the casing in the prototype stage, which can significantly reduce the time needed for 3D designing and printing. If it's possible to buy a good enough casing, the requirements: from R 2.2.5 to R 2.2.10 can all be satisfied. If in the prototype design, our team is behind schedule, we will very likely use this method for the casing.

7 Appendix: Test Plan

This section outlines the test plan to verify the design and outcome of the SyncLock DLA. Extensive testing will initially be carried out by the SyncLock team to verify the design, functionality, performance, and standards of the device, to ensure its usability and safety at all stages of development. In the proof-of-concept stage, testing will be performed in a laboratory, using function generators and power supplies to test its operational use and record potential errors for improvement. In the prototype stage, end-to-end testing will be performed on the device, as well as end-user testing, which will allow the customers to evaluate the product as a whole. A questionnaire will be administered so that feedback can be recorded for product improvement. A user guide will be available to let the testers know how to use the device.

The following questions will be posed, requiring a response on a scale of 1 to 10:

1. How easy was it to use the device?
2. Was there a struggle with adjusting the reference signal parameters?
3. Was it portable enough to take it to a field?
4. How safe did you feel when using the device?
5. Was the desired signal extracted by using this device?
6. Did the device meet your expectations?
7. How likely would you use this product?
8. Would you recommend this product?

7.1 Hardware Design Tests

ID: 7.1.1	**PoC Design Verification**	Name: Preamp Signal Process
Expectation	The amplitude of the signal passed through the preamp can be gained according to the value of gain we set and its signal is not distorted.	
Observation		

ID: 7.1.2	**Prototype Verification**	Name: Controlled Gain of Preamp
Expectation	Gain of preamp can be changed correctly by microprocessor.	
Observation		

ID: 7.1.3	**Prototype Verification**	Name: Save Output Data from DLA into SD Card
Expectation	The output data from DLA can be saved correctly in the SD card.	
Observation		

7.2 Digital Design Tests

ID: 7.2.1	**PoC Design Verification**	Name: Mixing Functionality Verification
Expectation	The data from the input signal and the reference signal is correctly multiplied together.	
Observation		

ID: 7.2.2	**PoC Design Verification**	Name: Filtering Functionality Verification
Expectation	The signal after mixing is correctly filtered by the low pass filter.	
Observation		

ID: 7.2.3	**PoC Design Verification**	Name: DDS Functionality Verification
Expectation	The reference signal can be correctly generated by the DDS according to the digital signals of the microprocessor.	
Observation		

ID: 7.2.4	**PoC Design Verification**	Name: Overall Functionality Verification
Expectation	The signal extracted from the output assembles the noise-buried input signal in amplitude and frequency.	
Observation		

7.3 Software Design Tests

ID: 7.3.1	**PoC Design Verification**	Name: Configure Amplifier & Reference Signals
Expectation	The amplitude, frequency, phase, and gain values can be set on the screen.	
Observation		

ID: 7.3.2	**PoC Design Verification**	Name: Configuration Error Handling
Expectation	Inputs of values outside the specified range will result in an error message, but does not break the program. Unplugging the device during configuration also displays an error message.	
Observation		

ID: 7.3.4	**Prototype Verification**	Name: Read and Save Output Data from DLA
Expectation	The output from the DAC can be read and displayed on the application. The output can also be downloaded and saved to a computer.	
Observation		

ID: 7.3.4	**Product Verification**	Name: Report DLA device faults
Expectation	The delay stores and displays faults recorded while in use. Records faults when plugged in and not plugged into a computer.	
Observation		

7.4 Questionnaire

Please answer all questions below with a tick, 1(Strongly Agree) to 10 (Strongly Disagree)												
S/N	**Questions**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**NA**
1	How easy was it to use the device?	○	○	○	○	○	○	○	○	○	○	○
2	Was there a struggle with adjusting the reference signal parameters?	○	○	○	○	○	○	○	○	○	○	○
3	Was it portable enough to take it to a field?	○	○	○	○	○	○	○	○	○	○	○
4	How safe did you feel when using the device?	○	○	○	○	○	○	○	○	○	○	○
5	Was the desired signal extracted by using this device?	○	○	○	○	○	○	○	○	○	○	○
6	Did the device meet your expectations?	○	○	○	○	○	○	○	○	○	○	○
7	How likely would you use this product?	○	○	○	○	○	○	○	○	○	○	○
8	Would you recommend this product?	○	○	○	○	○	○	○	○	○	○	○
Test Environment:												
Overall Reaction to DLA:												
Additional Comments:												