# Scuba Chat
# Project Update 5/3

Akshaya Bhat, Lilian Vu, Sienna Landa, Sophie Harris

# Project Overview

# Project Objective

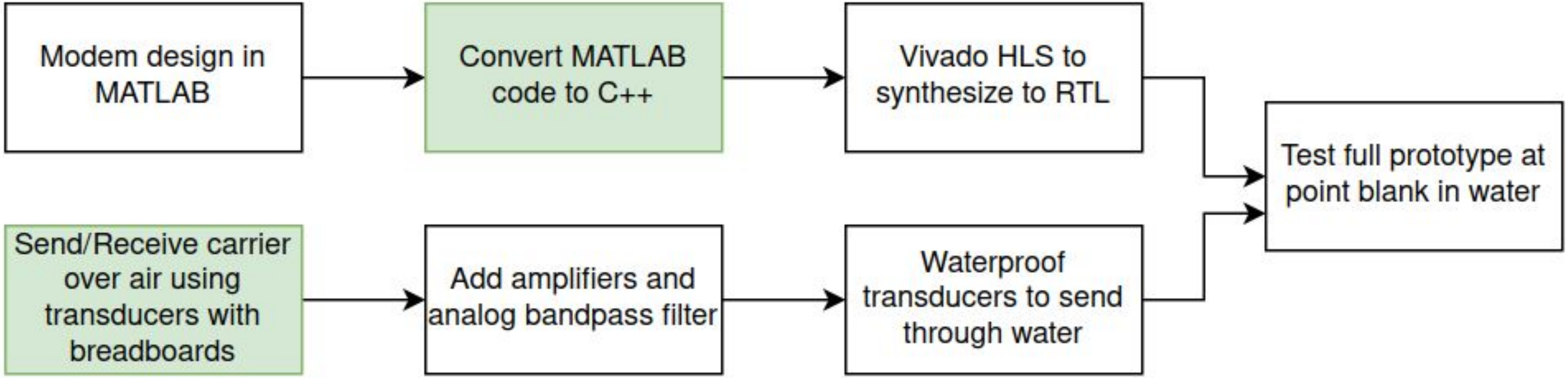
- Create an **underwater transmitter and receiver** using a piezoelectric transducer to generate ultrasound waves

- Implement the transceiver on our **PYNQ FPGAs** and **ARM cores**, with analog elements limited to the transducers, amplifiers, and a bandpass filter on the receiver side.

# Minimum Viable Product (MVP)

- **Basic Goals:**
  - Successfully **send** information over the underwater acoustic channel
  - **Prototype PYNQ boards** with waterproof transducer cabled to it that can be dipped several feet into the water or a swimming pool or lake to send data
  - Create **one Transmitter** and **one Receiver** for **one way** communication.
  - Have a high enough data rate to send small snippets of text **without noticeable delay**
  - **Bit error rate** should be around **1% ($10^{-2}$)**

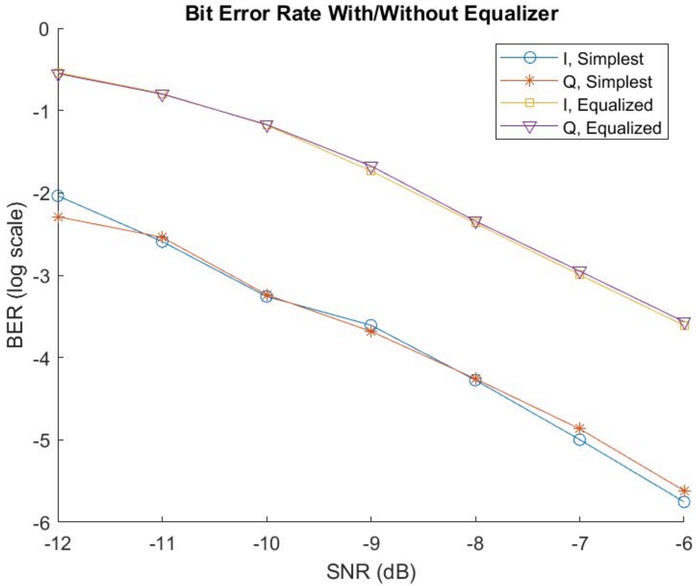
# Design Flow

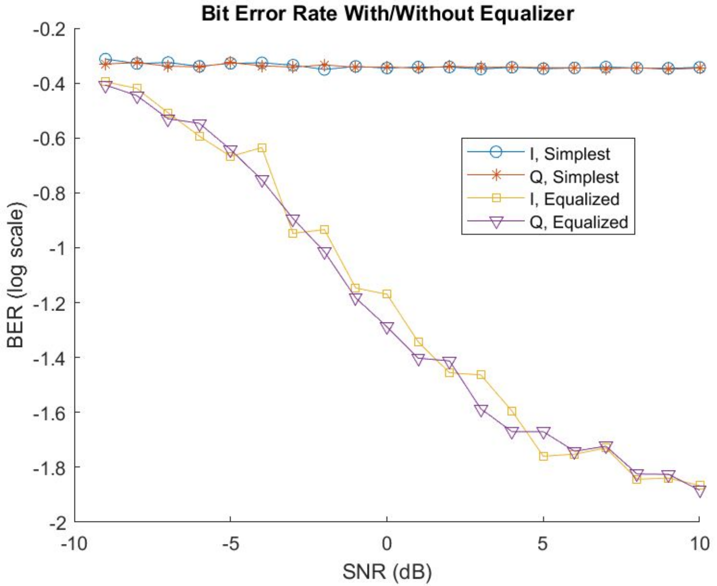

Modem design in MATLAB
Convert MATLAB code to C++
Vivado HLS to synthesize to RTL
Test full prototype at point blank in water
Send/Receive carrier over air using transducers with breadboards
Add amplifiers and analog bandpass filter
Waterproof transducers to send through water

🟩 Work In Progress

# Last Two Weeks

# BER Simulation Final Results (Sophie)





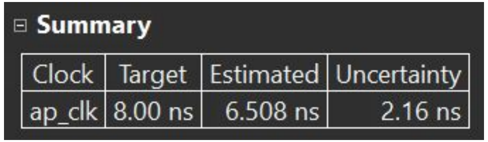
Summary: When we apply a severe multipath fading channel to our system, using channel equalization becomes critically important to correctly decoding information

# Transmitter Updates (Lilian)

- Moved Tx code to ARM core and handed it off to Akshaya to interface with DAC, transducer, and ADC
- Made C++ shell on ARM Core to interface
- Add in zeros for equalization

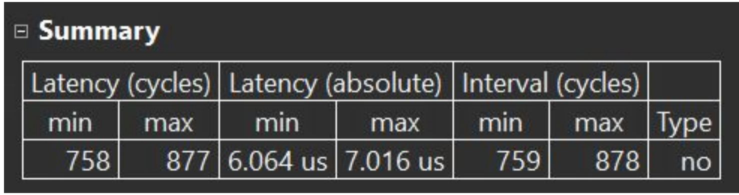
# Receiver - Upsampled (Sophie)

- Wrote the portion of the receiver which:
  - Demodulates
  - Applies matched filter
  - Identified packet start
  - Samples at optimal points 1 sample per symbol
- Written in C++, optimized for Vitis HLS
- Synthesized for PYNQ FPGA
- Meets timing requirements to sample at 128kHz from ADC
- Is correct within reasonable rounding error



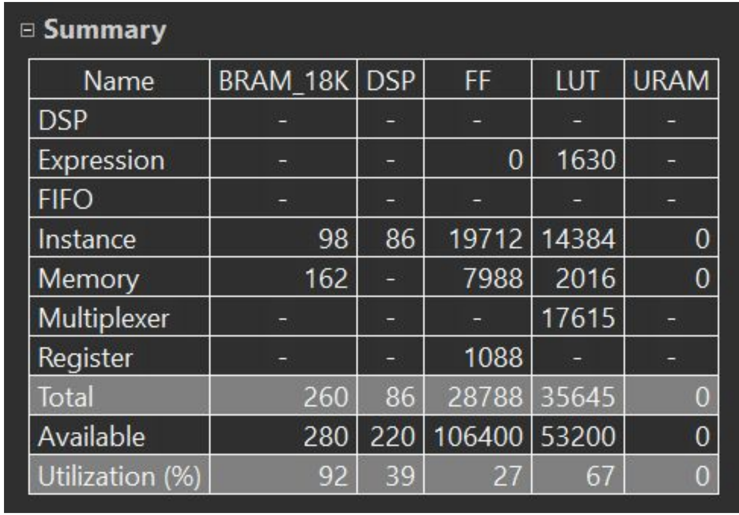
⊟ **Summary**

| Clock | Target | Estimated | Uncertainty |
|---|---|---|---|
| ap_clk | 8.00 ns | 6.508 ns | 2.16 ns |



⊟ **Summary**

| Latency (cycles) | | Latency (absolute) | | Interval (cycles) | | |
|---|---|---|---|---|---|---|
| min | max | min | max | min | max | Type |
| 758 | 877 | 6.064 us | 7.016 us | 759 | 878 | no |



⊟ **Summary**

| Name | BRAM_18K | DSP | FF | LUT | URAM |
|---|---|---|---|---|---|
| DSP | - | - | - | - | - |
| Expression | - | - | 0 | 1630 | - |
| FIFO | - | - | - | - | - |
| Instance | 98 | 86 | 19712 | 14384 | 0 |
| Memory | 162 | - | 7988 | 2016 | 0 |
| Multiplexer | - | - | - | 17615 | - |
| Register | - | - | 1088 | - | - |
| Total | 260 | 86 | 28788 | 35645 | 0 |
| Available | 280 | 220 | 106400 | 53200 | 0 |
| Utilization (%) | 92 | 39 | 27 | 67 | 0 |

# Receiver - Downsampled (Lilian)

- Halfway through C++ code after upsample portion of receiver code
- Channel equalization ✅
  - Toeplitz Matrix
  - PseudoInverse (SVD and Inverse)
- Viterbi decoder with soft decision decoding 🔄
- Descramble ✅
- Intending this portion of code to be on ARM Core

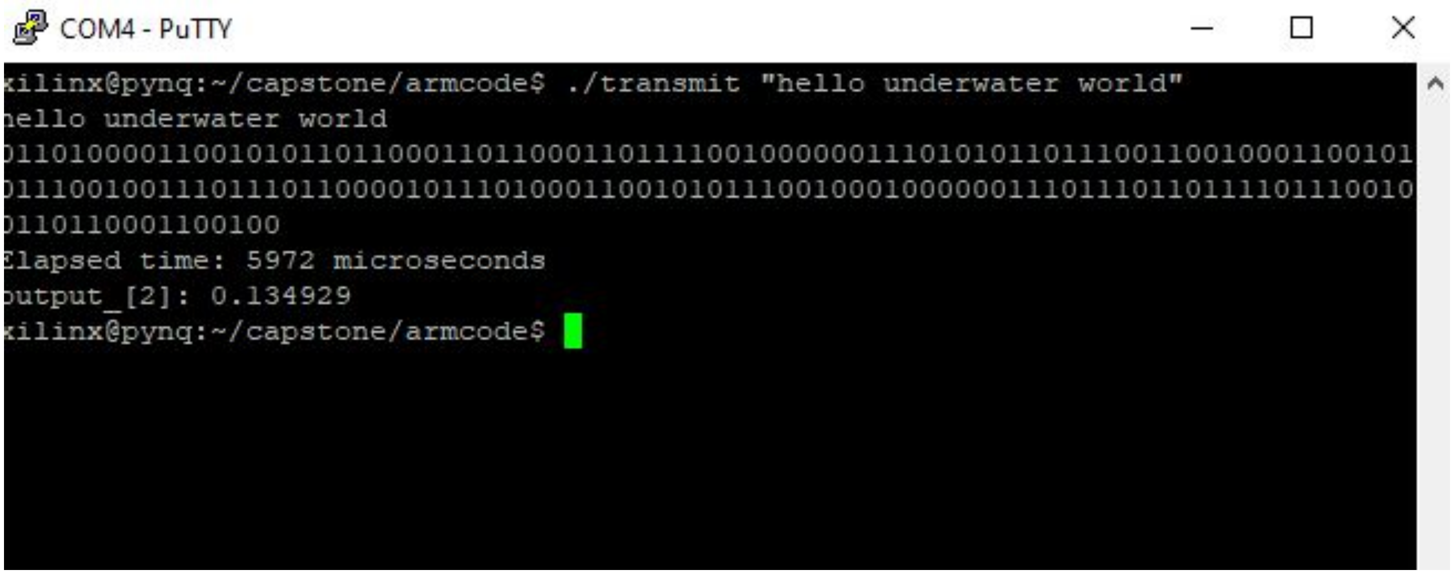
# Transmitter Hardware Updates (Akshaya)

- Created a user interface application for sending text messages to transmitter library (ARM)
    - Converts the text messages into binary and stores the modulated and encoded values in a file.
    - All negative values are transmitted as 0s and all positive values as 1s using GPIO on the jupyter notebook.
- Verified DAC to ADC loopback on PYNQ.

# Receiver Hardware Updates (Sienna)

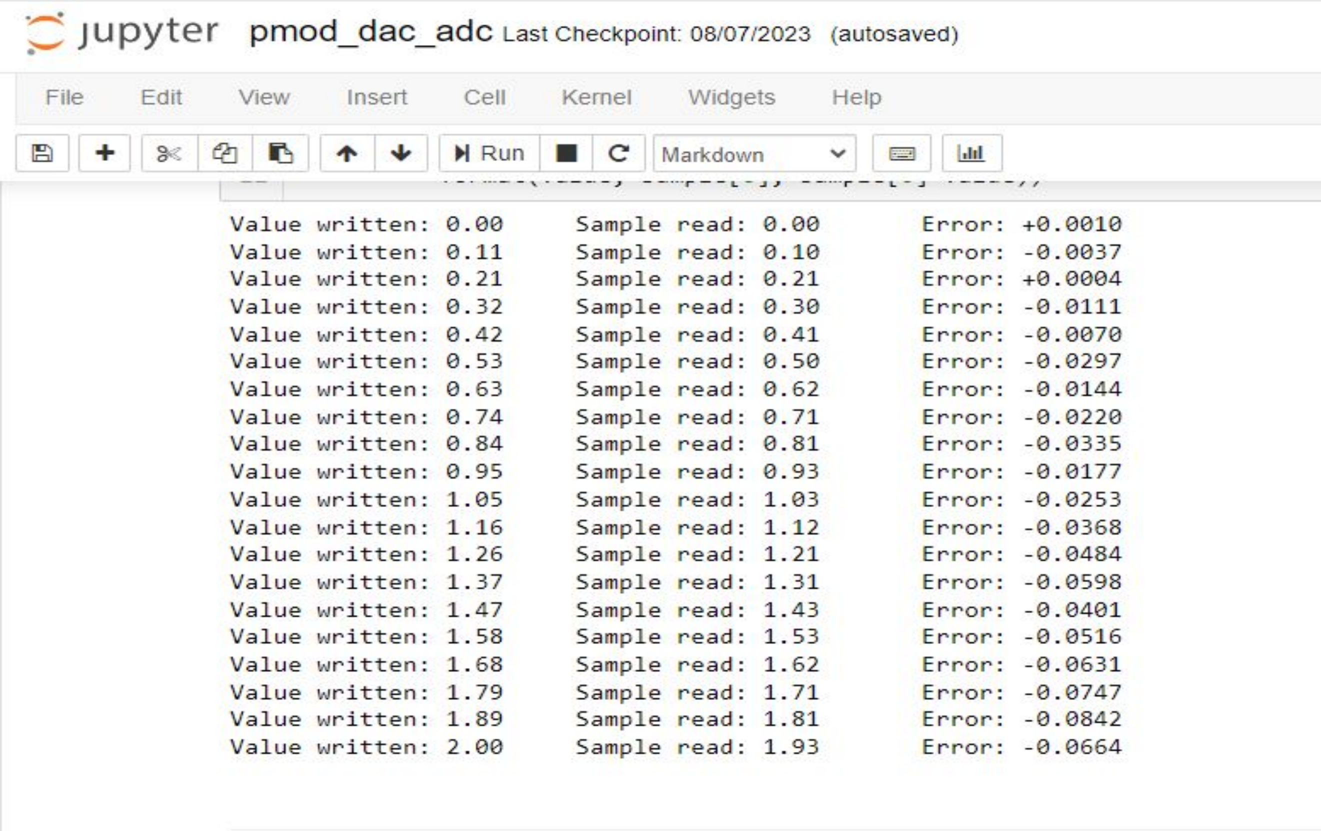
- Integrated Rx transducer with the PYNQ
    - Verified that it can detect signals from Tx Transducer circuit.
- Implemented analog bandpass filter and integrated it with the Rx circuit.

# Tx user interface application



```
xilinx@pynq:~/capstone/armcode$ ./transmit "hello underwater world"
hello underwater world
01101000011001010110110001101100011011110010000001110101011011100110010001100101
01110010011101110110000101110100011001010111001000100000011101110110111101110010
0110110001100100
Elapsed time: 5972 microseconds
output_[2]: 0.134929
xilinx@pynq:~/capstone/armcode$ 
```

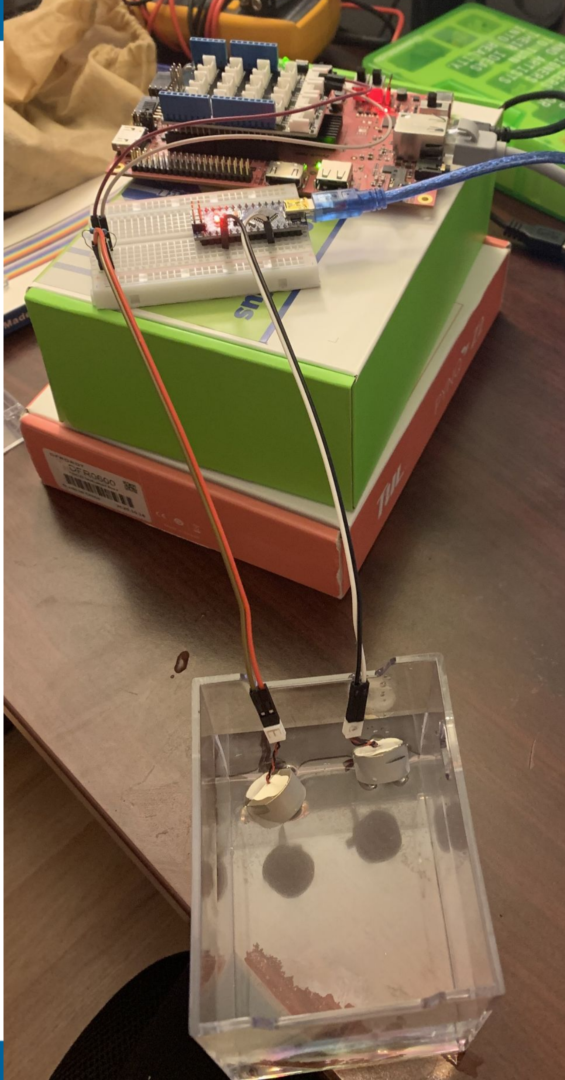
# DAC - to -ADC loopback



```
Value written: 0.00     Sample read: 0.00      Error: +0.0010
Value written: 0.11     Sample read: 0.10      Error: -0.0037
Value written: 0.21     Sample read: 0.21      Error: +0.0004
Value written: 0.32     Sample read: 0.30      Error: -0.0111
Value written: 0.42     Sample read: 0.41      Error: -0.0070
Value written: 0.53     Sample read: 0.50      Error: -0.0297
Value written: 0.63     Sample read: 0.62      Error: -0.0144
Value written: 0.74     Sample read: 0.71      Error: -0.0220
Value written: 0.84     Sample read: 0.81      Error: -0.0335
Value written: 0.95     Sample read: 0.93      Error: -0.0177
Value written: 1.05     Sample read: 1.03      Error: -0.0253
Value written: 1.16     Sample read: 1.12      Error: -0.0368
Value written: 1.26     Sample read: 1.21      Error: -0.0484
Value written: 1.37     Sample read: 1.31      Error: -0.0598
Value written: 1.47     Sample read: 1.43      Error: -0.0401
Value written: 1.58     Sample read: 1.53      Error: -0.0516
Value written: 1.68     Sample read: 1.62      Error: -0.0631
Value written: 1.79     Sample read: 1.71      Error: -0.0747
Value written: 1.89     Sample read: 1.81      Error: -0.0842
Value written: 2.00     Sample read: 1.93      Error: -0.0664
```

# Receiver Hardware Updates (Sienna)

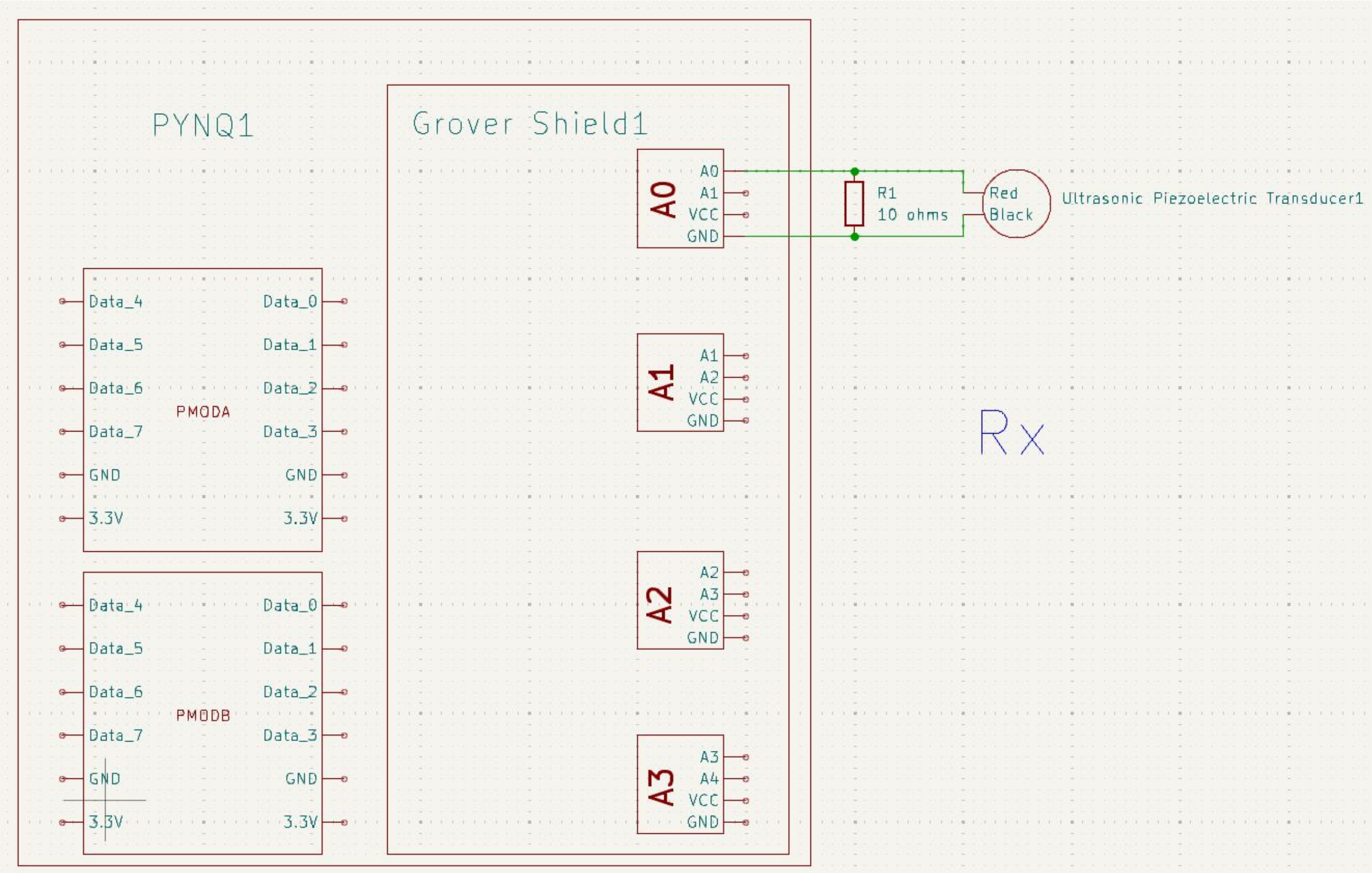
- ✓ **Integrated Rx** Transducer with **PYNQ**
- ✓ Implemented Analog Bandpass Filter
- ✓ Verified Rx Transducer on PYNQ is receiving Tx signals



# Receiver Hardware Updates (Sienna)

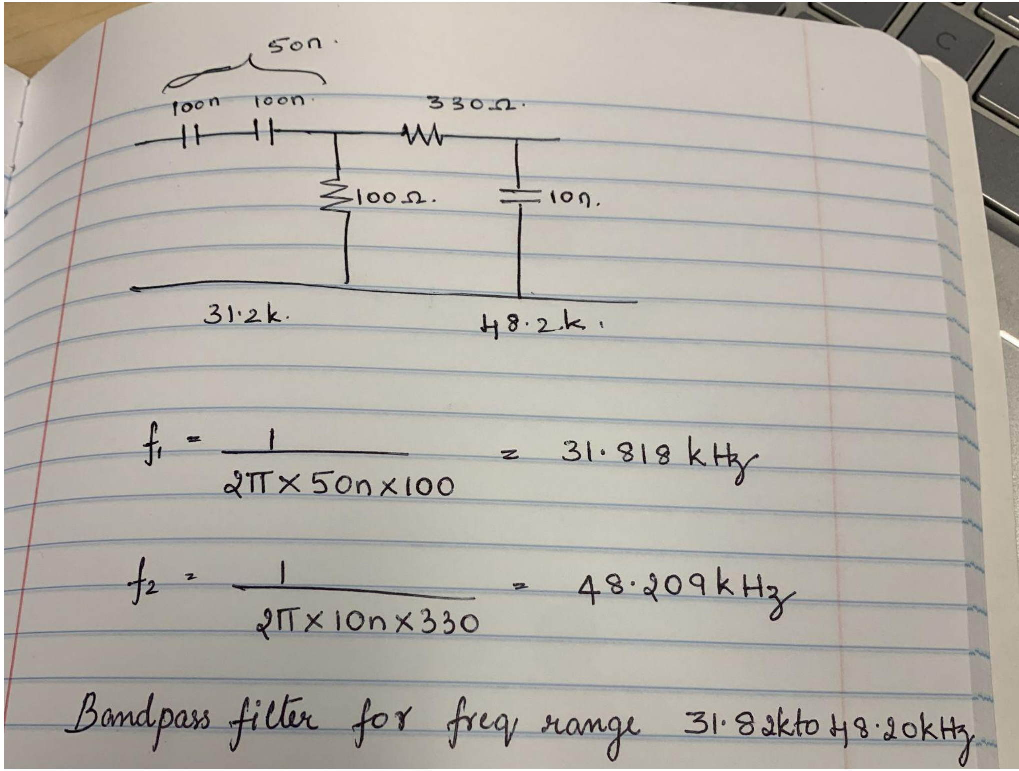
UC San Diego

- ✓ **Integrated Rx** Transducer with **PYNQ**
- ✓ Implemented Analog Bandpass Filter
- ✓ Verified Rx Transducer on PYNQ is receiving Tx signals



# Receiver Hardware Updates (Sienna)

- ✓ Integrated Rx Transducer with PYNQ
- ✓ Implemented Analog **Bandpass Filter**
- ✓ Verified Rx Transducer on PYNQ is receiving Tx signals



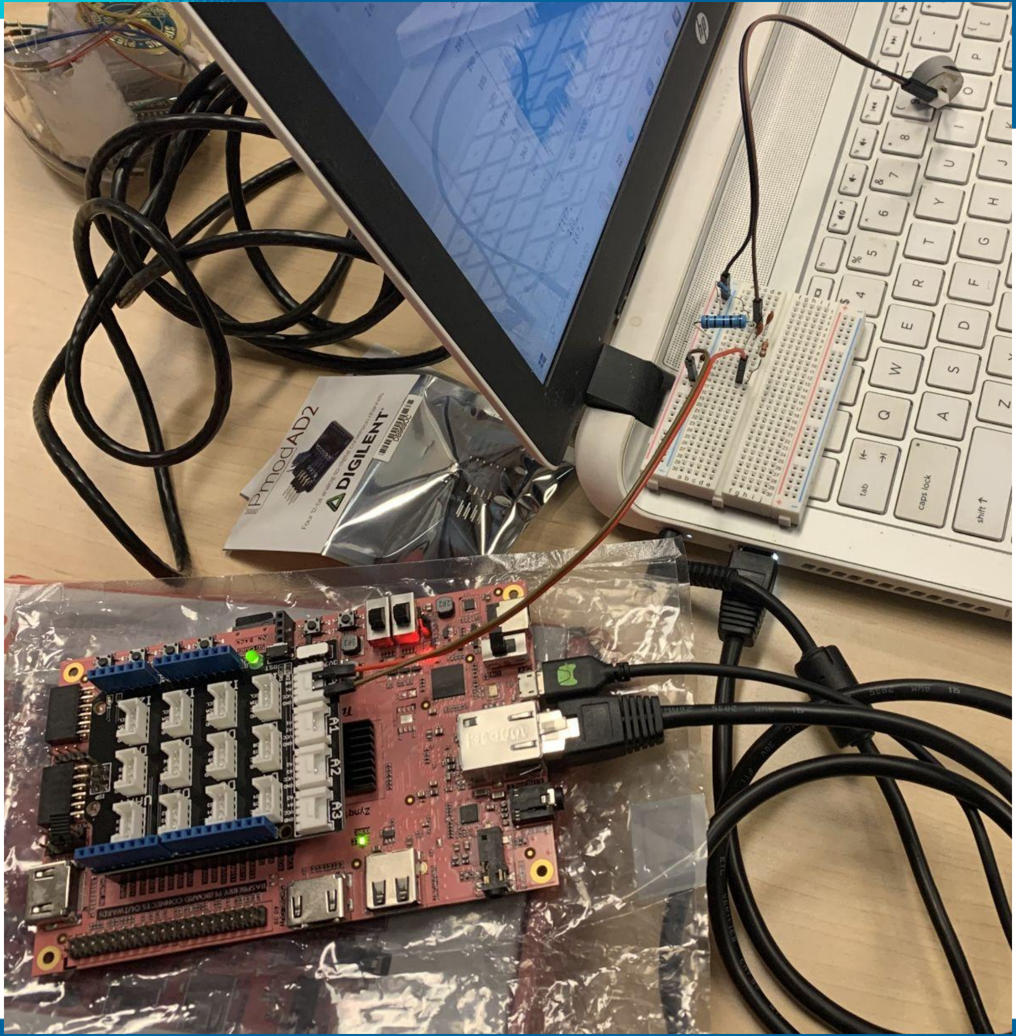
$$f_1 = \frac{1}{2\pi \times 50n \times 100} = 31{\cdot}818\ kHz$$

$$f_2 = \frac{1}{2\pi \times 10n \times 330} = 48{\cdot}209\ kHz$$

Bandpass filter for freq range 31·82k to 48·20kHz

# Receiver Hardware Updates

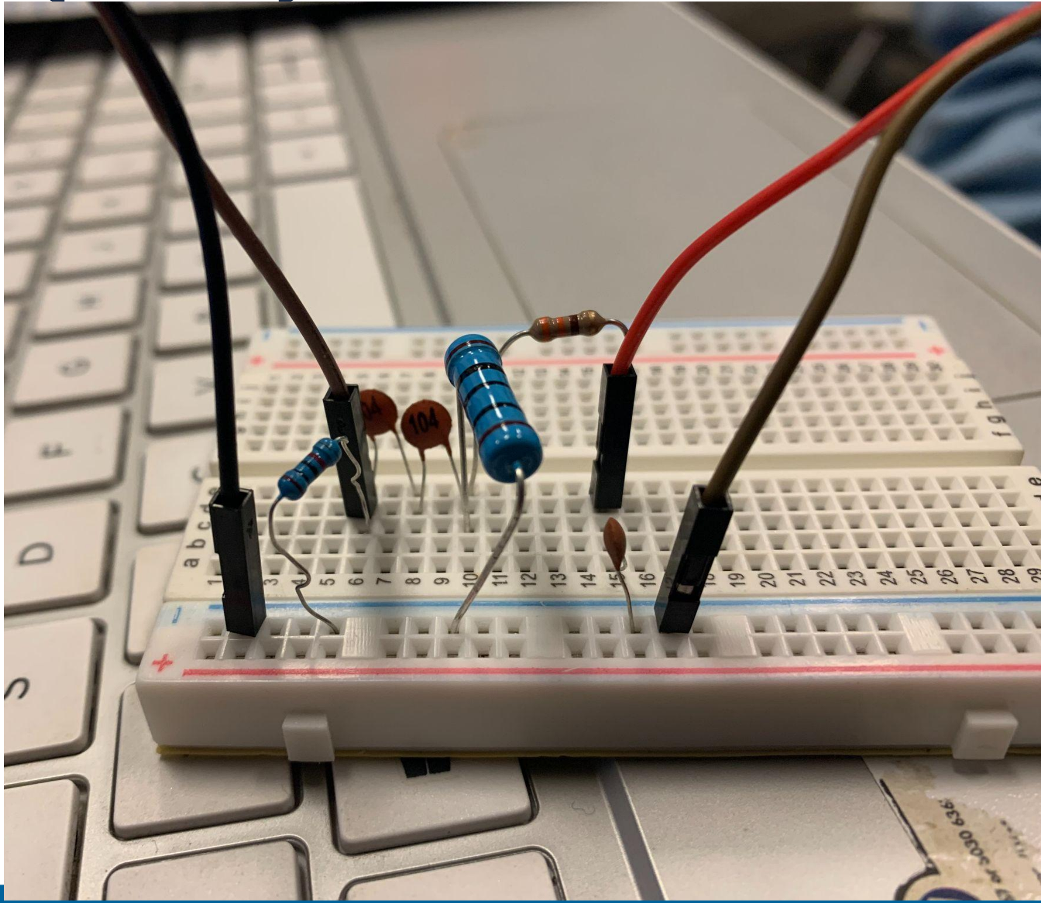
- ✓ Integrated Rx Transducer with PYNQ
- ✓ Implemented Analog **Bandpass Filter**
- ✓ Verified Rx Transducer on PYNQ is receiving Tx signals



# Receiver Hardware Updates (Sienna)

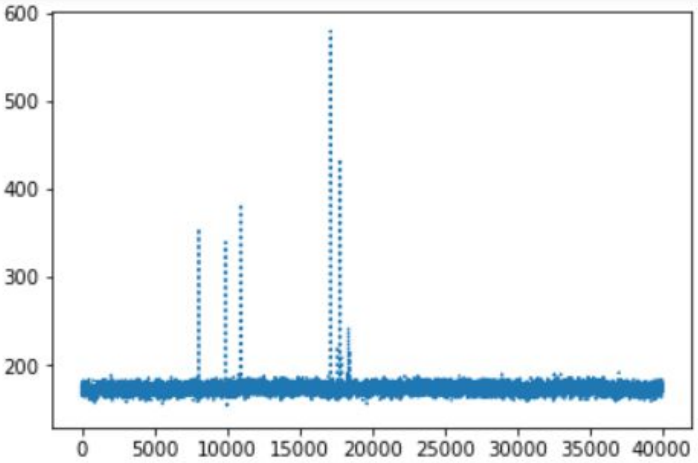
UC San Diego

- ✓ Integrated Rx Transducer with PYNQ
- ✓ Implemented Analog **Bandpass Filter**
- ✓ Verified Rx Transducer on PYNQ is receiving Tx signals



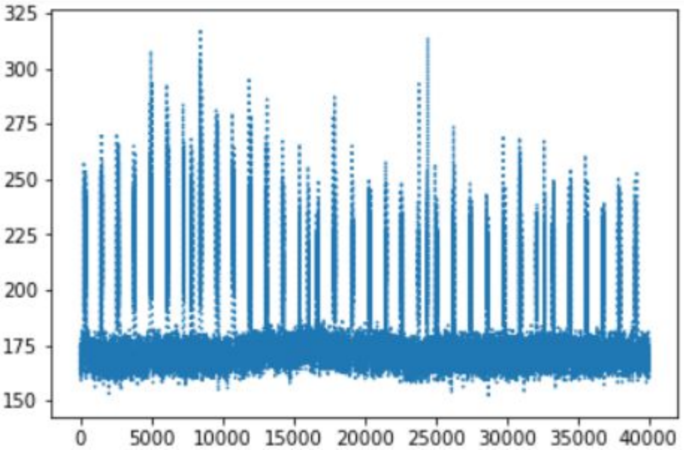
# Receiver Hardware Updates (Sienna)

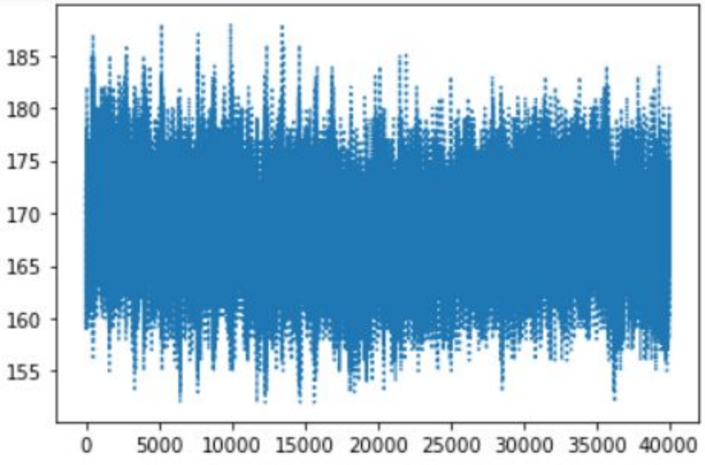
- ✓ Integrated Rx Transducer with PYNQ
- ✓ Implemented Analog Bandpass Filter
- ✓ **Verified Rx** Transducer on PYNQ is receiving Tx signals



Tapping Rx Transducer (w/ Finger)

Rx Transducer in water w/ Tx Transducer





Rx Transducer sitting in water

# Next Two Weeks

# Goals for the next 2 weeks

- Deadline - Task (Team member)
- 5/5 - Interface w/ DAC and ADC (Akshaya)
- 5/5 - Integrate ADC w/ Rx (Akshaya & Sophie)
- 5/7 - Implement Rx viterbi decoder (Lilian)
- 5/7 - Verify sine wave is Tx’ed and Rx’ed (Sienna)
- 5/11 - Test Bandpass Filter (Sienna)
- 5/11 - Integrate DAC w/ Tx (Akshaya)
- 5/16 - Implement Tx & Rx User Interface (Sienna)
- 5/16 - Determine if Automatic Gain Control (AGC) is needed (Akshaya)
- 5/16 - Integrate upsampled + downsample Rx and optimization (Lilian & Sophie)
- 5/16 - Test system from Tx to Rx (Everyone)

# ADC and Receiver interfacing (Akshaya/Sophie)

Software

- Routing for ADC to receiver FPGA block in Vivado
  - single input value per function call
  - Find appropriate threshold for identifying packet start (depends on ADC output values)
- Interface between FPGA receiver code and ARM core C++ receiver code
  - Flag to identify when packet values are ready for processing
  - Buffer containing the symbols themselves

Hardware

- Ensure we correctly capture sin waves using the ADC (correct reference voltages)
- Keep input voltage to ADC in safe range

# User Interface (Sienna)

## 1st Goal

- Transmit a random signal when a button is pressed
- When a signal is received …
  - Turn on a red LED for S.O.S.
  - Turn on a green LED for CLEAR

## 2nd Goal

- Display text messages on a LED screen

# Receiver code (Lilian)

- Implement C++ Viterbi Decoder - 5/7
- Merge upsampled receiver code with downsample code - 5/16
- Optimize and timing requirements
- Integrate on ARM Core with ADC and Sophie’s IP Core

# End to End Integration (Everyone)

- Start by connecting the transducers with vaseline to transmit with no channel
- Fix all of our interface issues between hardware/software and between software blocks
- Send a data packet and decode it on the receiver side
- We can do this in class because it does not require a large body of water

- Once it works, head out to the pool and try through water

- If it works with vaseline but not through water, our channel equalization is not good enough or doppler is getting us