Scuba Chat
Project Update 5/17

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)

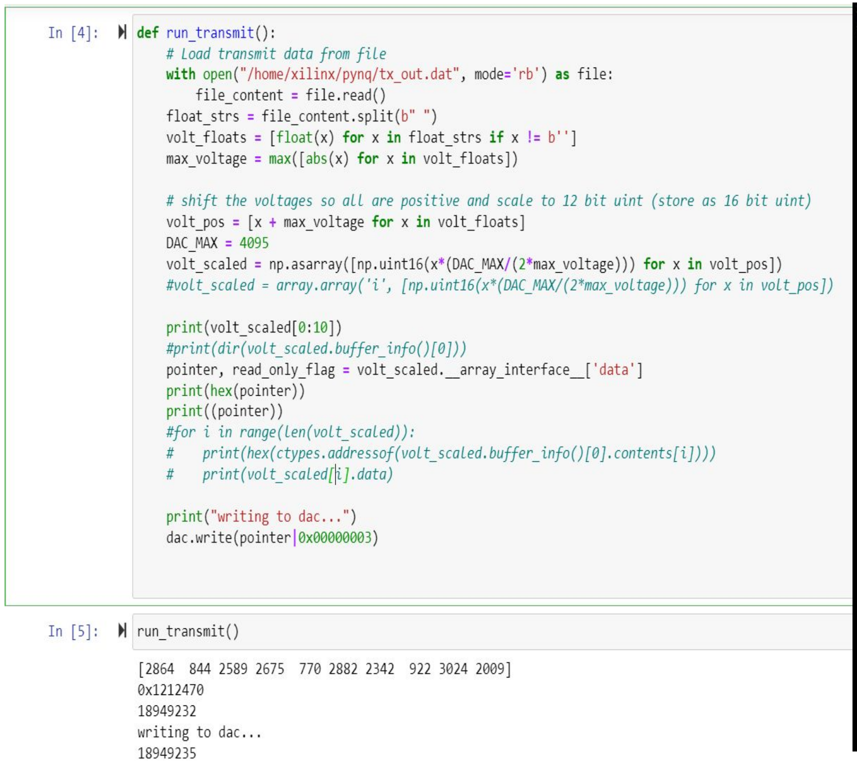
- **Basic Goals:**
    - Successfully **send** information over the underwater acoustic channel
    - **Prototype PYNQ boards** with waterproof transducer cabled to it that can be dipped several feet into the water or a swimming pool or lake to send data
    - Create **one Transmitter** and **one Receiver** 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})**

Last Two Weeks

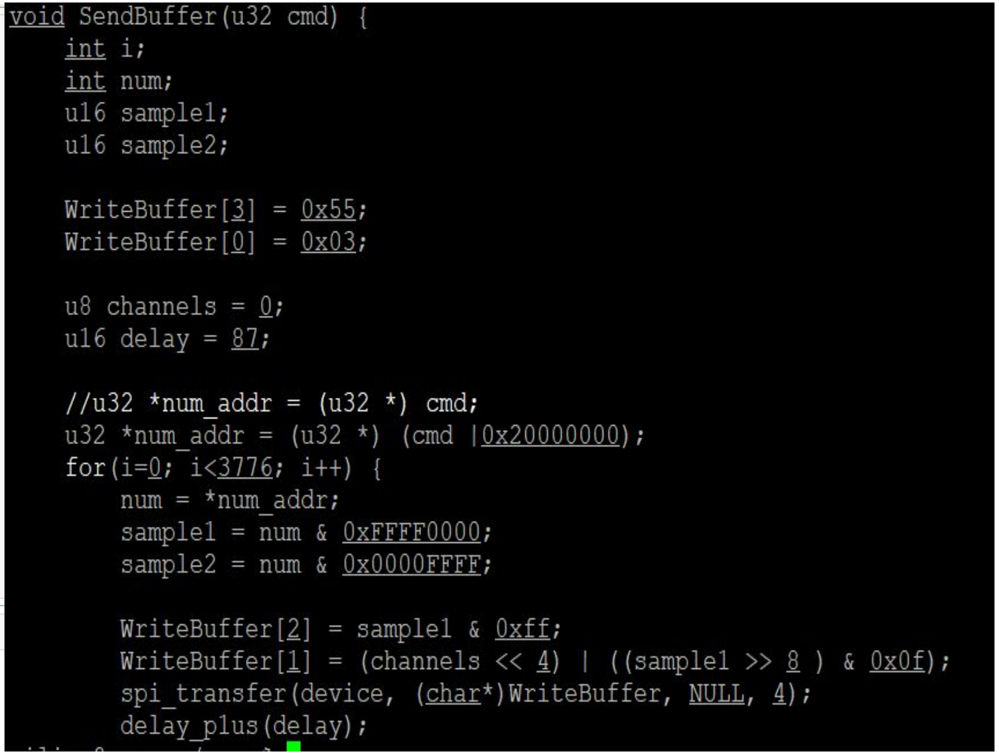
Tx Software

- Integration of TX ARM code with the DAC
    - Created a custom pmod_dac python file to read the output voltages
    - These voltages are passed to PMOD DAC with customized function to transmit over DAC
- Updated TX ARM code with configuration changes:
    - Reduce the oversample rate
    - Output samples to a file in order to read them to the DAC

Tx Software



```
In [4]:  def run_transmit():
             # Load transmit data from file
             with open("/home/xilinx/pynq/tx_out.dat", mode='rb') as file:
                 file_content = file.read()
             float_strs = file_content.split(b" ")
             volt_floats = [float(x) for x in float_strs if x != b'']
             max_voltage = max([abs(x) for x in volt_floats])

             # shift the voltages so all are positive and scale to 12 bit uint (store as 16 bit uint)
             volt_pos = [x + max_voltage for x in volt_floats]
             DAC_MAX = 4095
             volt_scaled = np.asarray([np.uint16(x*(DAC_MAX/(2*max_voltage))) for x in volt_pos])
             #volt_scaled = array.array('i', [np.uint16(x*(DAC_MAX/(2*max_voltage))) for x in volt_pos])

             print(volt_scaled[0:10])
             #print(dir(volt_scaled.buffer_info()[0]))
             pointer, read_only_flag = volt_scaled.__array_interface__['data']
             print(hex(pointer))
             print((pointer))
             #for i in range(len(volt_scaled)):
             #    print(hex(ctypes.addressof(volt_scaled.buffer_info()[0].contents[i])))
             #    print(volt_scaled[i].data)

             print("writing to dac...")
             dac.write(pointer|0x00000003)

In [5]:  run_transmit()

         [2864  844 2589 2675  770 2882 2342  922 3024 2009]
         0x1212470
         18949232
         writing to dac...
         18949235
```



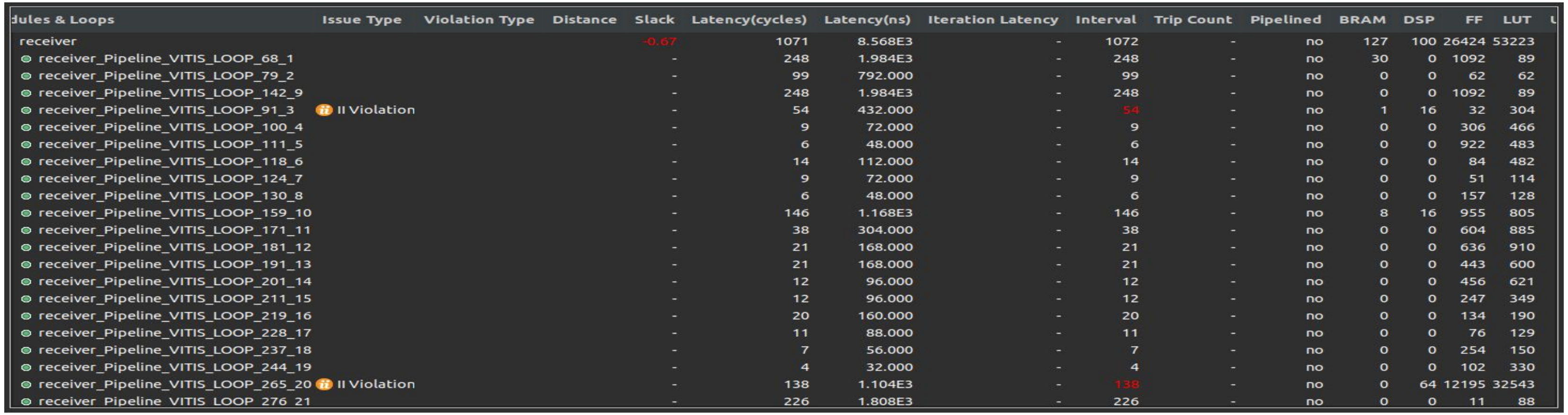
```
void SendBuffer(u32 cmd) {
    int i;
    int num;
    u16 sample1;
    u16 sample2;

    WriteBuffer[3] = 0x55;
    WriteBuffer[0] = 0x03;

    u8 channels = 0;
    u16 delay = 87;

    //u32 *num_addr = (u32 *) cmd;
    u32 *num_addr = (u32 *) (cmd |0x20000000);
    for(i=0; i<3776; i++) {
        num = *num_addr;
        sample1 = num & 0xFFFF0000;
        sample2 = num & 0x0000FFFF;

        WriteBuffer[2] = sample1 & 0xff;
        WriteBuffer[1] = (channels << 4) | ((sample1 >> 8 ) & 0x0f);
        spi_transfer(device, (char*)WriteBuffer, NULL, 4);
        delay_plus(delay);
```

Rx Software

- Implemented upsampled portion of the receiver for FPGA
  - In process of integrating with ADC
- Implemented downsampled portion of the receiver for ARM
  - Descrambling, decoding, channel equalization
  - Used someone else’s viterbi decoder code because it worked better than what we were doing: https://github.com/williamyang98/ViterbiDecoderCpp/tree/master
  - Communicates with upsampled code via a DMA

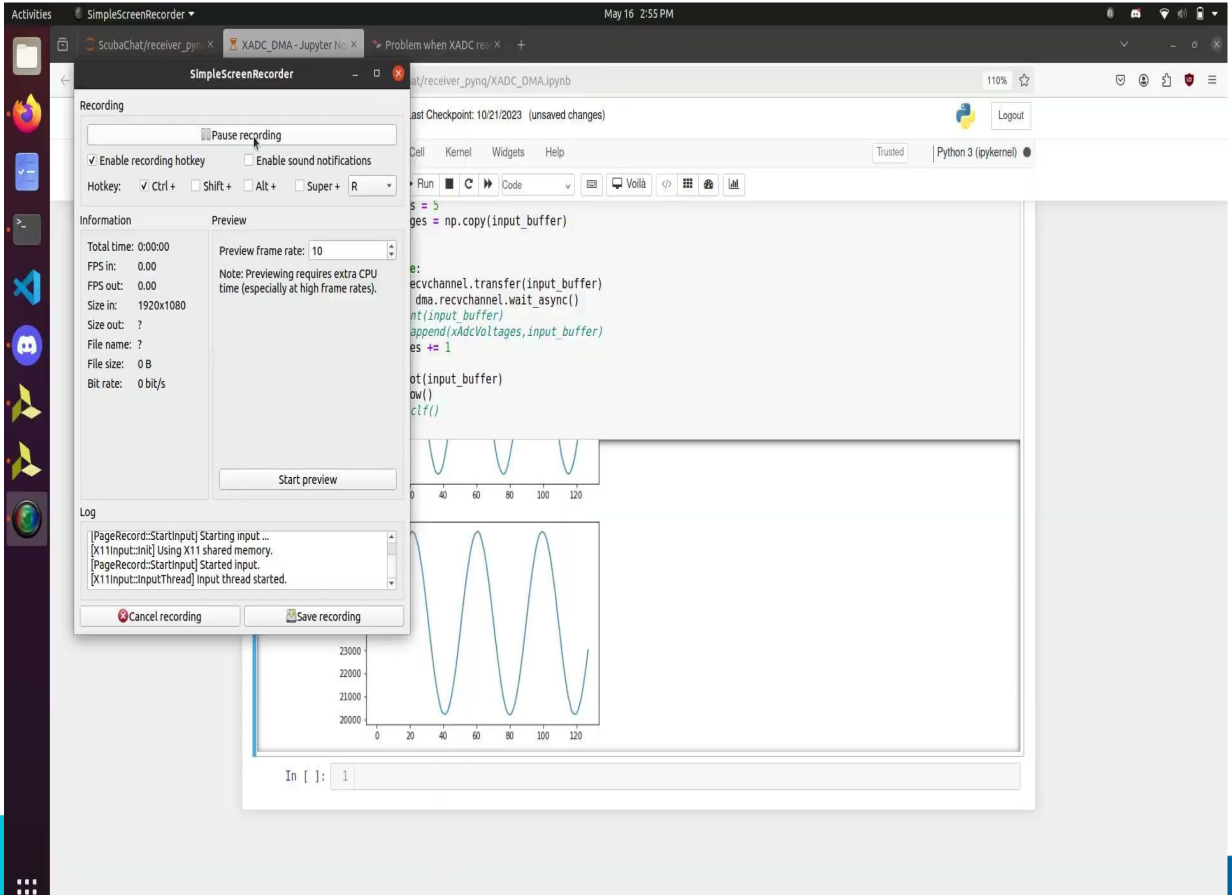
Rx Upsampled

- Connecting ADC to DMA to Custom IP Block through FPGA
  - Using XADC Wizard
  - Make sure IP meets resource utilization and timing requirements
- Can receive ADC data and pass to IP block



dules & Loops	Issue Type	Violation Type	Distance	Slack	Latency(cycles)	Latency(ns)	Iteration Latency	Interval	Trip Count	Pipelined	BRAM	DSP	FF	LUT
receiver				-0.67	1071	8.568E3	-	1072	-	no	127	100	26424	53223
receiver_Pipeline_VITIS_LOOP_68_1				-	248	1.984E3	-	248	-	no	30	0	1092	89
receiver_Pipeline_VITIS_LOOP_79_2				-	99	792.000	-	99	-	no	0	0	62	62
receiver_Pipeline_VITIS_LOOP_142_9				-	248	1.984E3	-	248	-	no	0	0	1092	89
receiver_Pipeline_VITIS_LOOP_91_3	II Violation			-	54	432.000	-	54	-	no	1	16	32	304
receiver_Pipeline_VITIS_LOOP_100_4				-	9	72.000	-	9	-	no	0	0	306	466
receiver_Pipeline_VITIS_LOOP_111_5				-	6	48.000	-	6	-	no	0	0	922	483
receiver_Pipeline_VITIS_LOOP_118_6				-	14	112.000	-	14	-	no	0	0	84	482
receiver_Pipeline_VITIS_LOOP_124_7				-	9	72.000	-	9	-	no	0	0	51	114
receiver_Pipeline_VITIS_LOOP_130_8				-	6	48.000	-	6	-	no	0	0	157	128
receiver_Pipeline_VITIS_LOOP_159_10				-	146	1.168E3	-	146	-	no	8	16	955	805
receiver_Pipeline_VITIS_LOOP_171_11				-	38	304.000	-	38	-	no	0	0	604	885
receiver_Pipeline_VITIS_LOOP_181_12				-	21	168.000	-	21	-	no	0	0	636	910
receiver_Pipeline_VITIS_LOOP_191_13				-	21	168.000	-	21	-	no	0	0	443	600
receiver_Pipeline_VITIS_LOOP_201_14				-	12	96.000	-	12	-	no	0	0	456	621
receiver_Pipeline_VITIS_LOOP_211_15				-	12	96.000	-	12	-	no	0	0	247	349
receiver_Pipeline_VITIS_LOOP_219_16				-	20	160.000	-	20	-	no	0	0	134	190
receiver_Pipeline_VITIS_LOOP_228_17				-	11	88.000	-	11	-	no	0	0	76	129
receiver_Pipeline_VITIS_LOOP_237_18				-	7	56.000	-	7	-	no	0	0	254	150
receiver_Pipeline_VITIS_LOOP_244_19				-	4	32.000	-	4	-	no	0	0	102	330
receiver_Pipeline_VITIS_LOOP_265_20	II Violation			-	138	1.104E3	-	138	-	no	0	64	12195	32543
receiver_Pipeline_VITIS_LOOP_276_21				-	226	1.808E3	-	226	-	no	0	0	11	88

Receiver - Demo (Sig Gen - ADC)



SimpleScreenRecorder

Recording

Pause recording

[✓] Enable recording hotkey   [ ] Enable sound notifications

Hotkey: [✓] Ctrl + [ ] Shift + [ ] Alt + [ ] Super + R

Information

Total time: 0:00:00
FPS in: 0.00
FPS out: 0.00
Size in: 1920x1080
Size out: ?
File name: ?
File size: 0 B
Bit rate: 0 bit/s

Preview

Preview frame rate: 10

Note: Previewing requires extra CPU time (especially at high frame rates).

Start preview

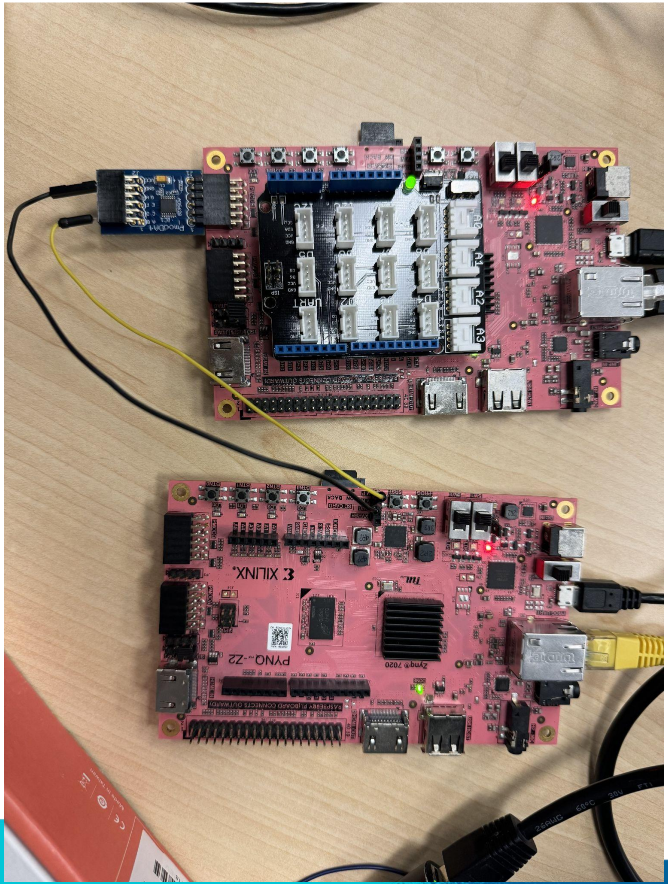
Log

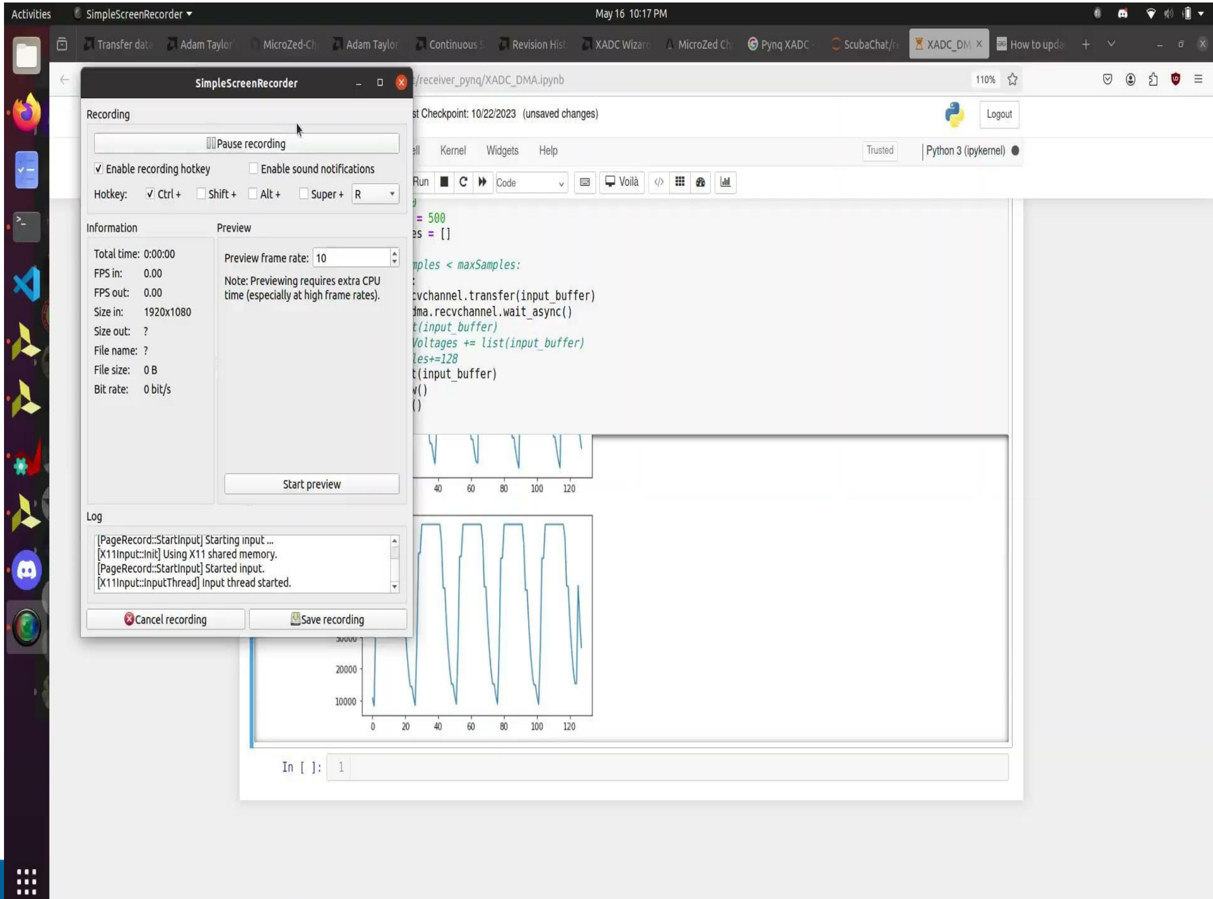
```
[PageRecord::StartInput] Starting input ...
[X11Input::Init] Using X11 shared memory.
[PageRecord::StartInput] Started input.
[X11Input::InputThread] Input thread started.
```

Cancel recording   Save recording

```
s = 5
ges = np.copy(input_buffer)


e:
ecvchannel.transfer(input_buffer)
 dma.recvchannel.wait_async()
nt(input_buffer)
append(xAdcVoltages,input_buffer)
es += 1

ot(input_buffer)
ow()
clf()
```

Tx-Rx Square Wave Demo (DAC-ADC)





st Checkpoint: 10/22/2023 (unsaved changes)

SimpleScreenRecorder

Recording

Pause recording

☑ Enable recording hotkey ☐ Enable sound notifications

Hotkey: ☑ Ctrl + ☐ Shift + ☐ Alt + ☐ Super + R

Information

Total time:	0:00:00
FPS in:	0.00
FPS out:	0.00
Size in:	1920x1080
Size out:	?
File name:	?
File size:	0 B
Bit rate:	0 bit/s

Preview

Preview frame rate: 10

Note: Previewing requires extra CPU time (especially at high frame rates).

Start preview

Log

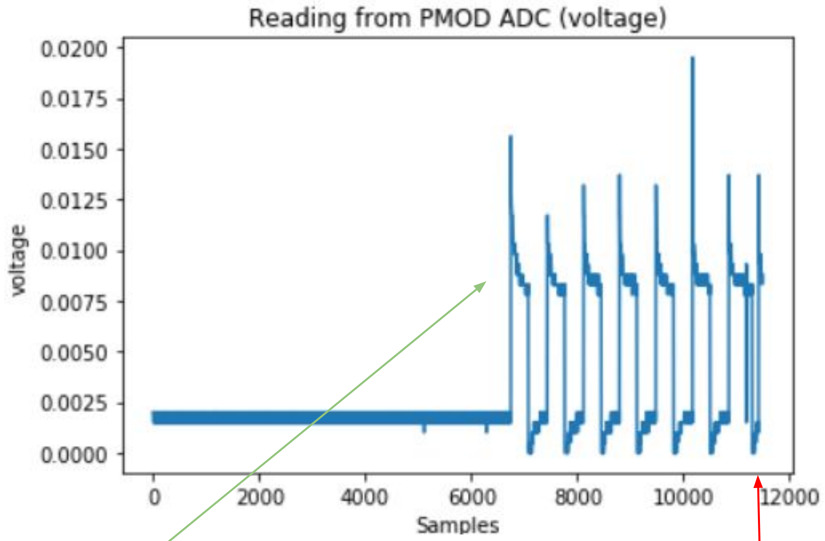
```
[PageRecord::StartInput] Starting input ...
[X11Input::Init] Using X11 shared memory.
[PageRecord::StartInput] Started input.
[X11Input::InputThread] Input thread started.
```

Cancel recording Save recording

```
= 500
es = []

mples < maxSamples:
:
cvchannel.transfer(input_buffer)
dma.recvchannel.wait_async()
t(input_buffer)
Voltages += list(input_buffer)
les+=128
t(input_buffer)
w()
()
```

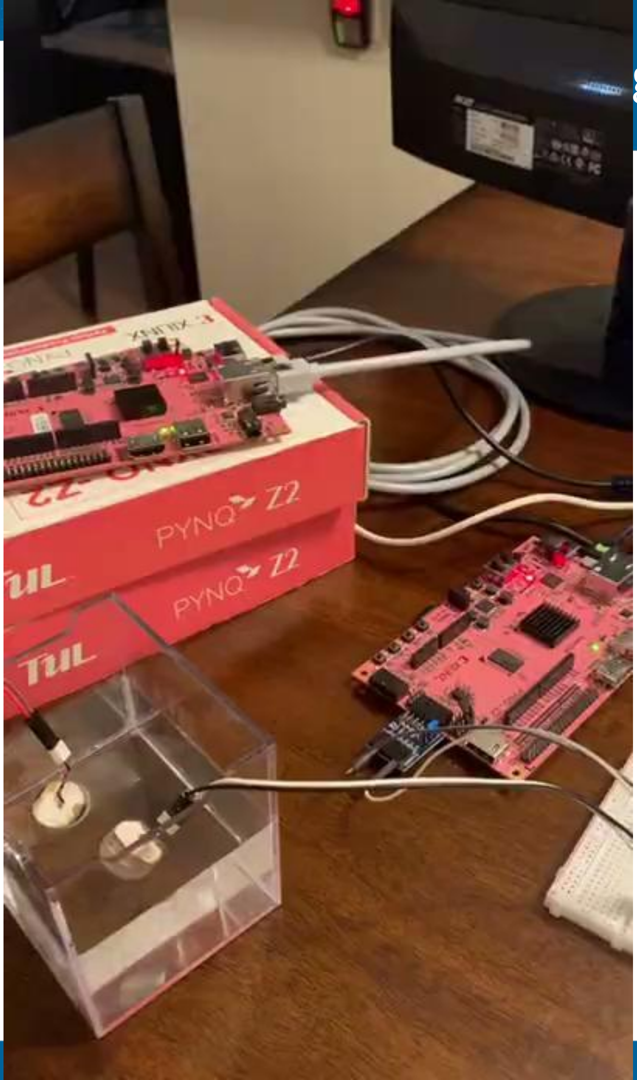
In [ ]: 1

Tx-Rx Buttons Demo



Green Button Pressed

Red Button Pressed



Goals for the last 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)

Simon Peter Sacramento Landa

Next Two Weeks

UC San Diego

Goals for the next 2 weeks

- Interface DAC and ADC with transducer and amplifier
- Integrate ADC + upsampled IP + downsample ARM
- Test Bandpass Filter
- Integrate user interface with Tx and Rx
- Verify system from Tx to Rx

Verification Plan

1. Verify downsampled tx/rx
   - Pass the results of the downsampled tx directly to the downsampled rx
2. Verify entire tx/rx software
   - Pass the results of the tx software into the rx software
3. Verify software + DMAs, DAC and ADC
   - Wire the DAC and ADC together and send data end to end
4. Field test: Verify end to end with transducers through water
   - Test in a pool, dip the transducers in water and send data

Problems to Solve

- Limits in range of ADC sampling frequency, we can’t run at our desired sampling rate
- DMA doesn’t like to run taking in 128 samples at a time and outputting 1 at a time
- DAC seems to decrease voltage value too slowly after write