EE 492 Biweekly Report 12

3/29/21 – 4/12/21

Group Number: SD May 21-43

Project Title: Emergency! Need backup!

Client/Advisor: Collins Aerospace / Andrew Bolstad

Team Members / Role:

James Curtis / Meeting Scribe

Caroline Easley / Meeting Facilitator

Marcelo Abrantes / Engineer (Power Systems)

Michael Kuehn / Communications Director

Benjamin Welte / Project Documentation

Abbey Wilder / Test Engineer

Stepan Zelenin / Engineer (Communication Systems)

Period Summary:

The main task occupying the receiver and transmitter design teams during the past work period was the finalization of the schematic and layout for the final PCB that will carry the radio circuit. This entailed finishing the part selection for our project as well as the routing on the circuit card.

Regarding SNMP control of the radio, we successfully finished parametrizing the code to control the local oscillator and resolved our previous issue with the amplitude and shape of the local oscillator output by realizing that the oscillator's output is a differential signal – after we measured it as such, its output matched our expectations. We will still need to deploy the code to control the local oscillator onto the new part that we ordered because the oscillator that we had been testing does not cover the full frequency range that the radio needs to transmit and receive at.

We also began integrating the SNMP user interface into the local network in the TLA which presented several complications that were not present when it was being developed at one of our team members' homes. Deploying the SNMP code in the TLA required us to acquire IP addresses for our Arduinos using DHCP, and we also needed to connect to the same router using physical ethernet connections (previously the SNMP interface had worked over Wi-Fi, but the network configuration in the TLA necessitated that the computer running MIB browser be physically connected to the router like the Arduino).

In short, we believe that we have all of the pieces in place to begin testing our final product when the PCB arrives. We have already verified the schematic for the radio via simulation and simply need to confirm that the hardware works as expected with some final integration testing.

Past Period Accomplishments:

- Finalized parts selection for the PCB's voltage regulators, VHF and UHF band filters, and multiplexors – Marcelo
- Finished the PCB schematic and layout for the final product – Caroline, James, Michael, Stepan, and Marcelo
- Finalized linear regulator selection to generate the final PCB's 5V and 3.3V supply voltages – Michael
- Deployed SNMP user interface on the local network in the TLA – Ben, Abbey
- Configured IP addresses for the Arduinos using DHCP protocol – Ben
- Finished parametrizing the code to program the local oscillator to output a square wave of arbitrary frequency – Ben
- Redesigned mixers for the final PCB to better adhere to project requirements using pre-input and post-output signal conditioning chain adjustments (this entailed adding resistors for impedance matching and a ferrite bead for biasing via a balun center tap) – Stepan
- Finalized the BOM for all of the PCB's parts – Caroline, Michael
- Ordered the BOM and PCB via the ETG – Michael
- Verified final schematic using SPICE simulations – Stepan

Pending Issues:

- Make SNMP user interface compile on Windows as well as on Mac OS – Ben, Abbey
- Solder new local oscillator to dev board and verify its operation – Everybody
- Test the final PCB when it arrives – Marcelo, Caroline, James, Michael, Stepan

Individual Contributions:

Name	Individual Contributions	Hours this week	Hours cumulative
James C.	• Reviewed final PCB layout • Helped configure SNMP on the LAN in the TLA	12	66
Caroline E.	• Reviewed PCB layout and schematic • Finalized BOM with the PCB's final parts	15	71
Marcelo A.	• Finalized PCB schematic, layout, & part selection (voltage regulators, filters for VHF/UHF Bands, and multiplexors)	20	81
Michael K.	• Finalized PCB schematic and part selection (specifically the 5V and 3.3V linear regulators) • Integrated voltage regulators into the schematic	24	78

	• Finalized the PCB's filter design • Finalized BOM and ordered the final PCB & parts list from ETG		
Ben W.	• Finalized Arduino code to program the local oscillator • Acquired IP addresses for the Arduinos using DHCP • Resolved issues with the shape and amplitude of the old oscillator's output waveform • Began transferring code for the old oscillator to the new oscillator • Successfully deployed SNMP user interface on the TLA's LAN	18	79
Abbey W.	• Successfully deployed SNMP user interface on the TLA's LAN	12	69
Stepan Z.	• Redesigned mixer biasing to better fulfill project requirements • Adjusted mixer pre-input and post-output signal conditioning	24	82

	chain using impedance matching with resistors and a ferrite bead • Finalized differential audio-range filter design. • Performed concluding tests of schematic design in SPICE • Finalized schematic & layout for the final PCB		

Plans for the Upcoming Period:

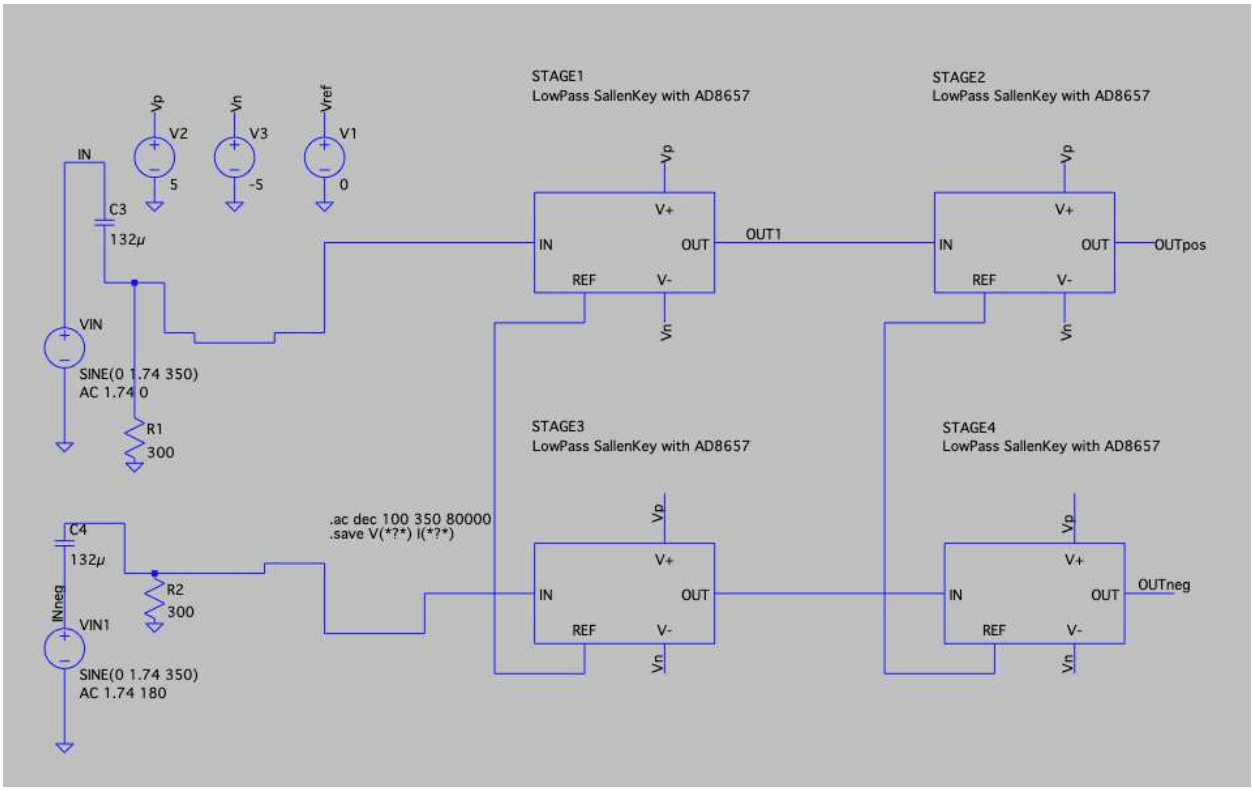
During the upcoming work period, we plan to begin testing our final PCB when it arrives. Hopefully, we will be able to resolve any issues that we discover without ordering a new board; otherwise, we will simply have to explain in our presentation why our product isn't fully functional and recommend fixes for future engineering teams to implement. We also need to integrate our SNMP interface with the code to program the local oscillator now that both have been successfully deployed in the TLA, and we need to ensure that the code for the local oscillator works on the new part that we ordered from Silicon Labs because the local oscillator that we ordered for testing doesn't cover the full frequency range over which our radio needs to send and receive.

Advisor Meeting Summary:

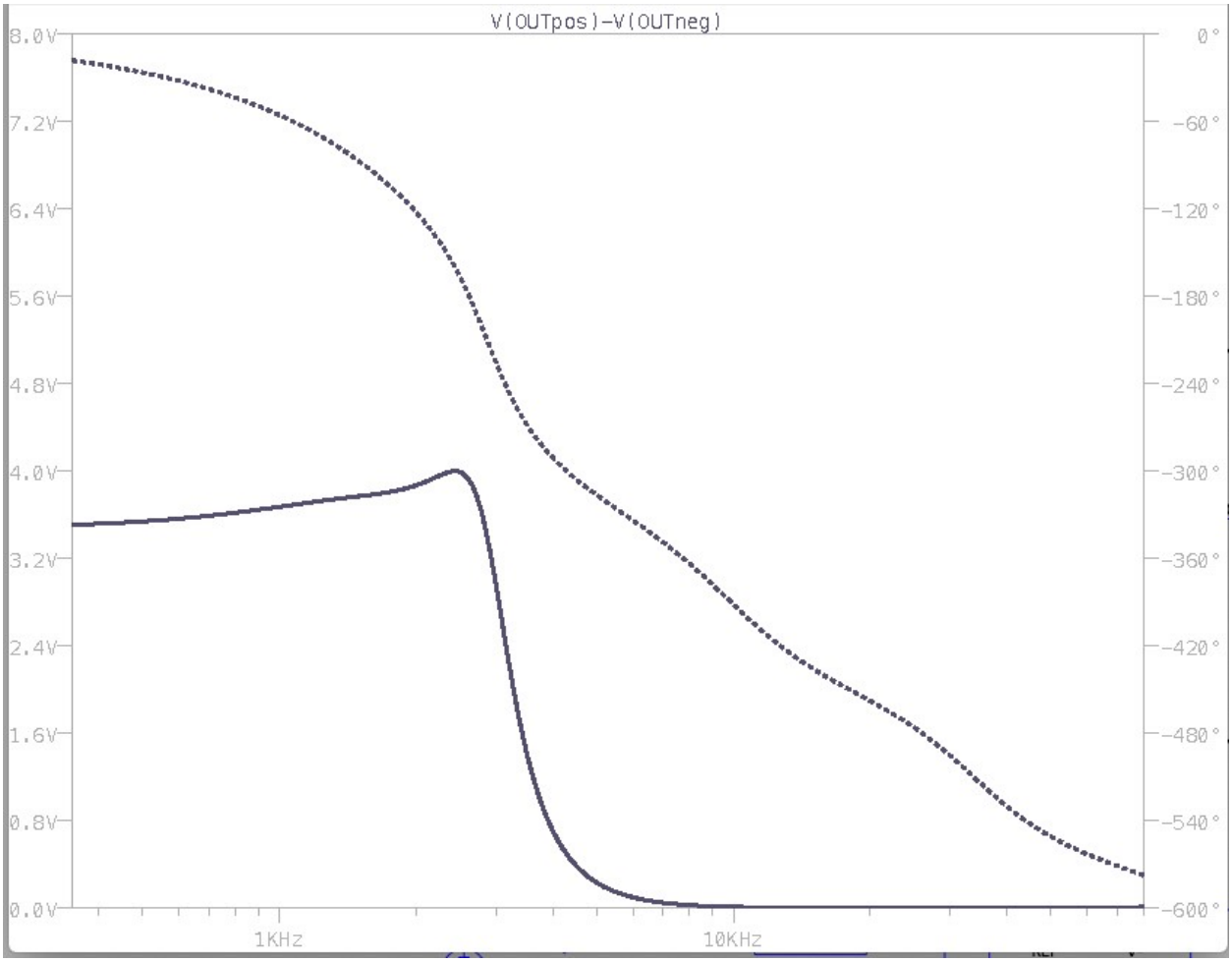
During our meetings with Dr. Bolstad, we discussed the difficulties we were experiencing while integrating our SNMP interface with the local network in the TLA and finalizing the design of our PCB. His main recommendation was to reserve lab space ahead of time to ensure that we have the resources needed to finish the project before the deadline for our presentation. He also recommended that we assemble a brief presentation to show to our client during our next

meeting with them in order to update them on our progress, so we began collecting images associated with our work in order to present them to Collins on 4/13.

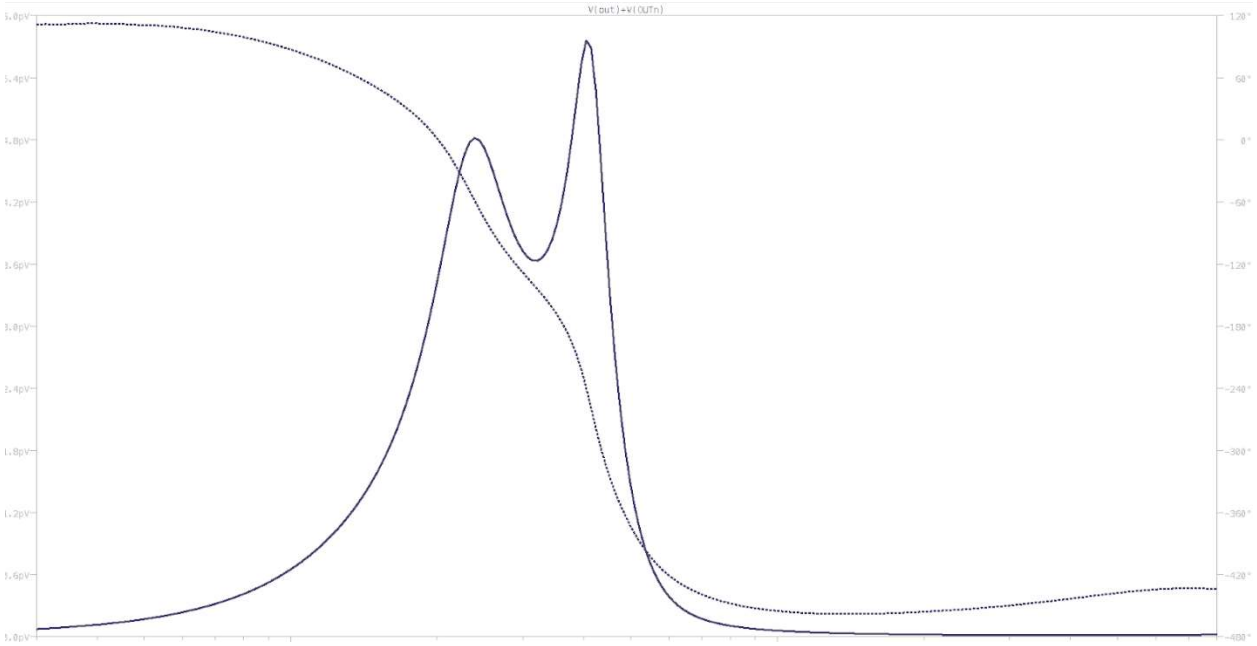
Appendix A: Images



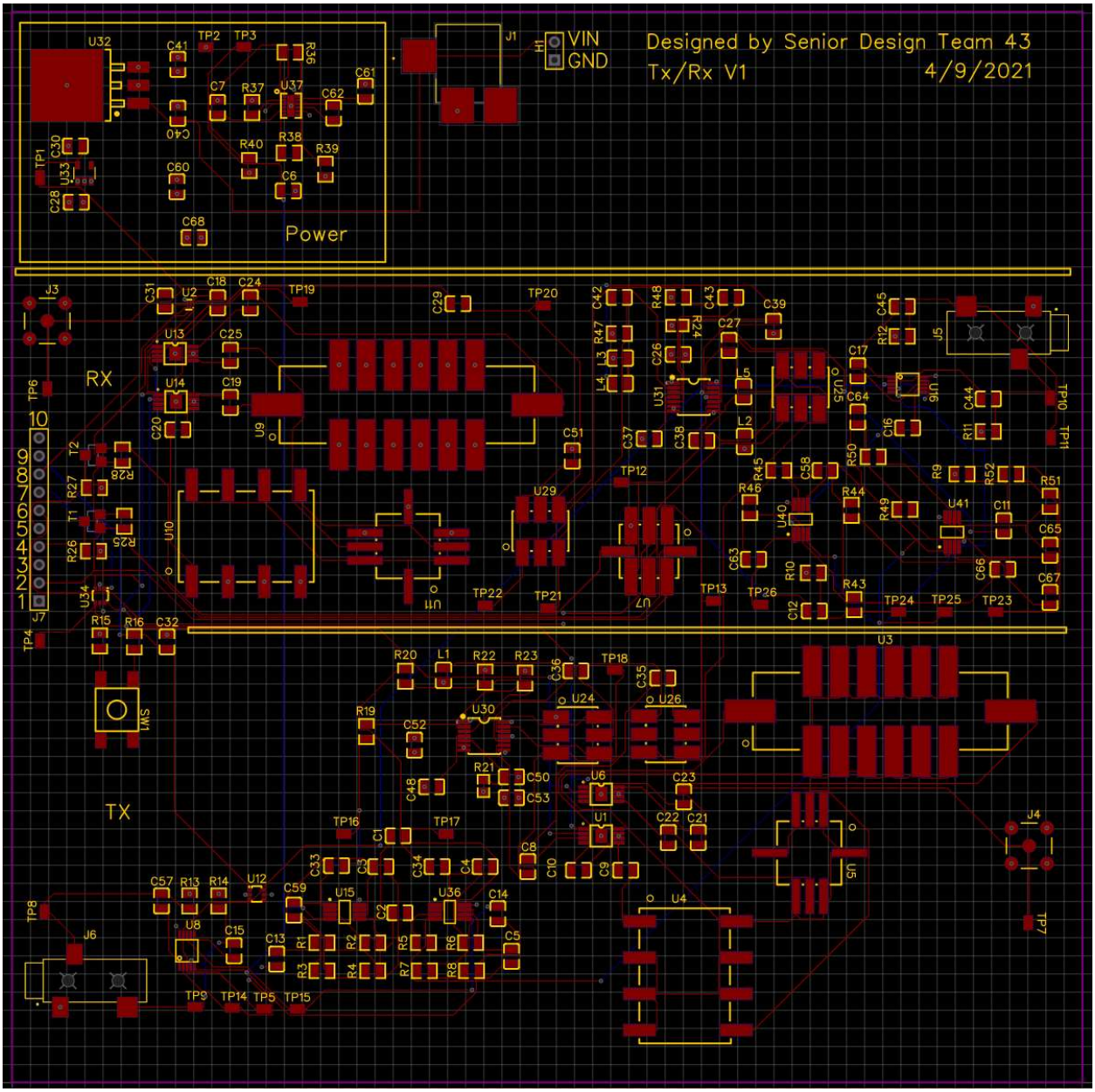
schematic for low-pass differential audio range filter



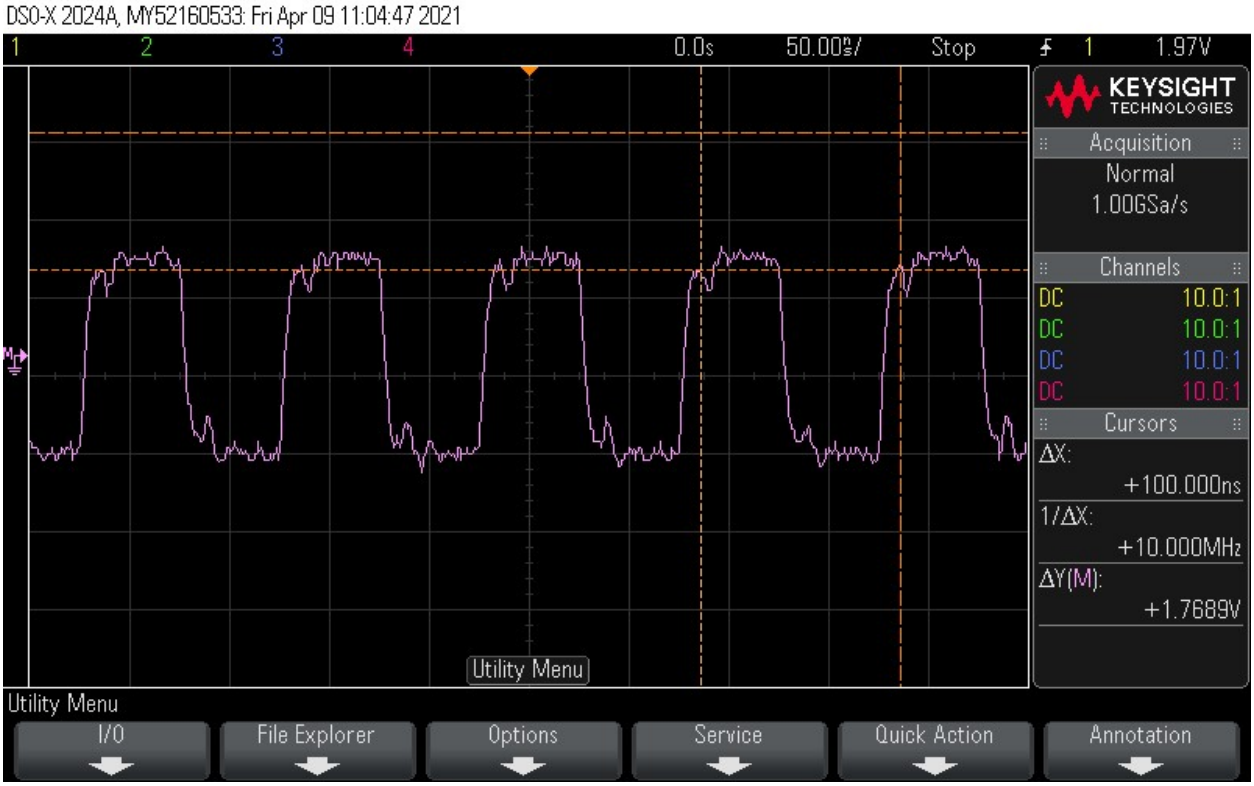
audio filter’s frequency response (magnitude and phase, differential mode)



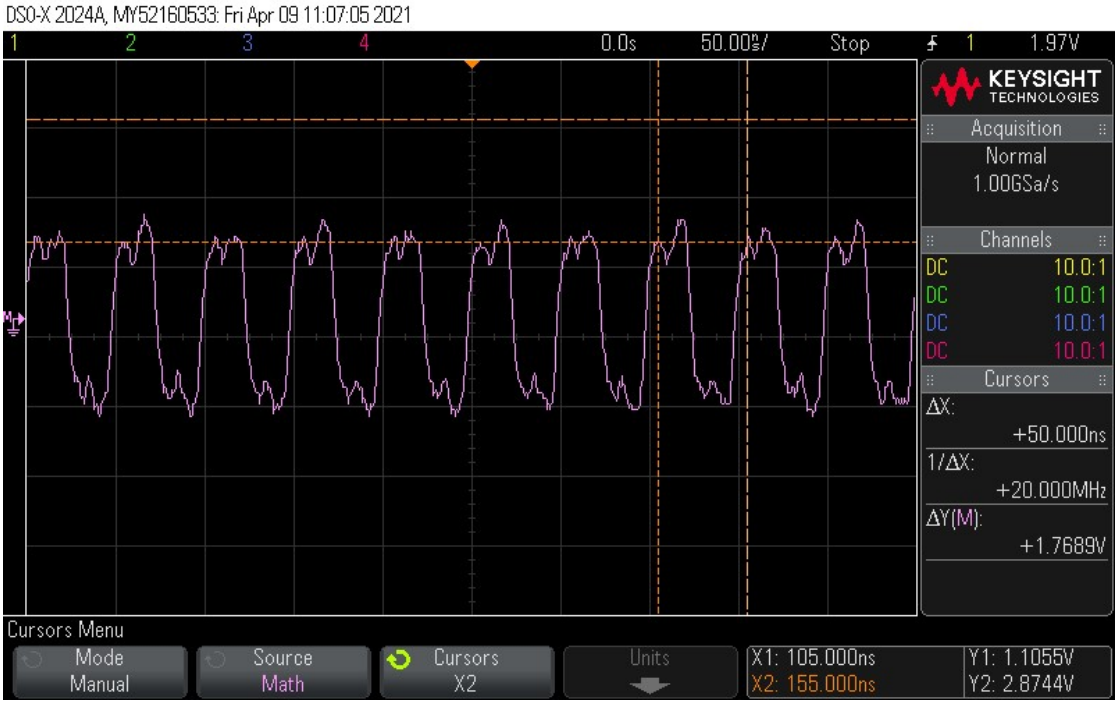
audio filter’s frequency response (magnitude and phase, common mode)



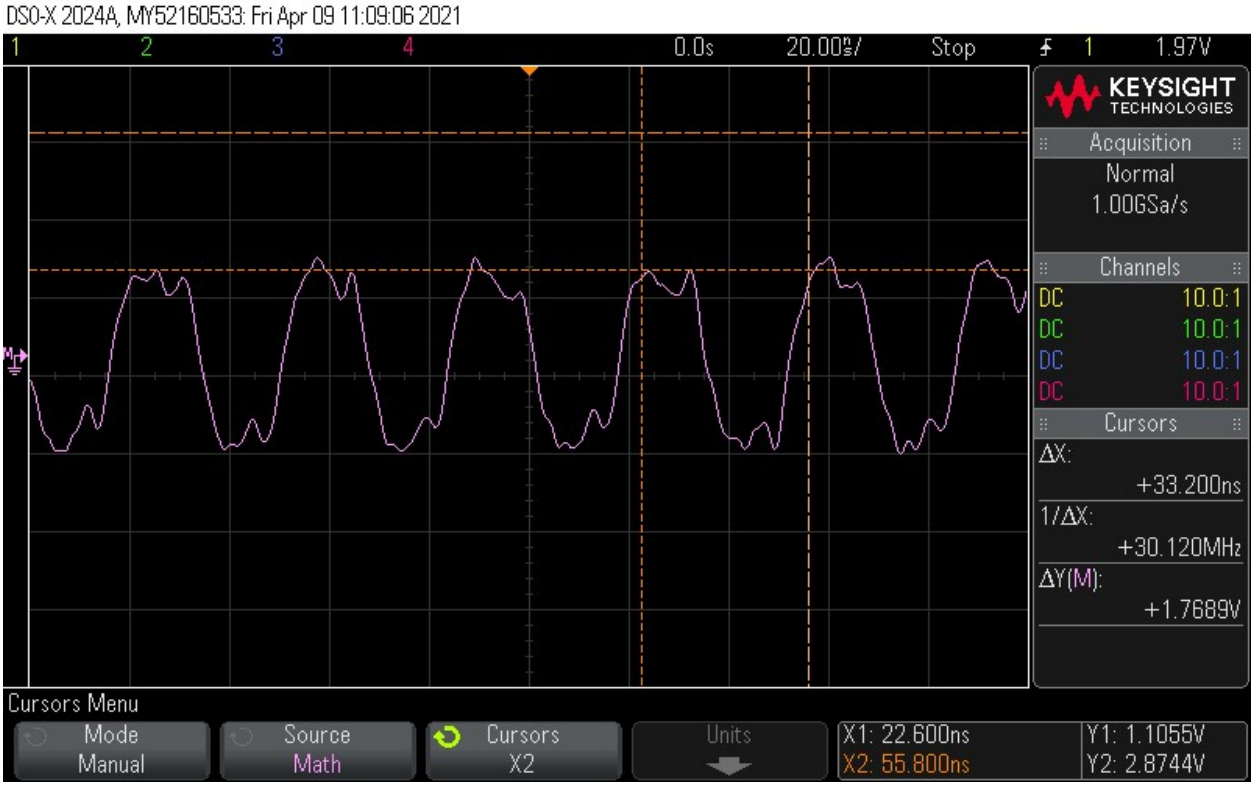
Final PCB Layout



Old local oscillator's differential output (10 MHz)



Old local oscillator's differential output (20 MHz)



Old local oscillator's differential output (30 MHz)

Appendix B: Code

Header file for functions to program the local oscillator (LO.h):

```
#ifndef _LO_H
#define _LO_H
typedef unsigned long uint32_t;

/*
 * Gen_Params
 *  - Description:  function to generate the N1 and HS_DIV parameters given a certain frequency
 *
 *  - Inputs:       F_Params[] - the uninitialized array of Freq. parameters (F_Params[0] is N1, and F_Params[1] is HS_DIV)
 *
 *  - Outputs:      nothing, but it fills the F_Params array
 *
 *
 *    NOTES: 5.67 GHz > (F * HS_DIV * N1) > 4.86 GHz
 *           valid values of N1: 1, 2, 4, 6, etc ...
 *           valid values of HS_DIV: 4, 5, 6, 7, 9, 11
 */
void Gen_Params(int F_Params[], double Freq);

 /*
 * hardcode_test
 *  - Description:  test function to get LO behavior when reg values are hard-coded
 *
 *    NOTE: register values are specific to the local oscillator used for initial testing
 */
void hardcode_test();
```

```
/*
* Reset_LO
*  - Description:  reset the local oscillator to 10 MHz
*
*    NOTE: register values are specific to the local oscillator used for initial testing
*/
void Reset_LO();


/*
* Read_LO_Config
*  - Description:  read LO configuration registers for debugging
*/
void Read_LO_Config();


/*
* Write_LO_Values
*  - Description:  Helper function to write the values of HS_DIV, N1, and RFREQ to the appropriate LO registers
*
*  - Inputs:       N1_reg_val - the value for N1 to be written to LO registers 7 [4:0] and 8 [7:6]
*                HS_DIV_reg_val - the value for HS_DIV to be written to  LO register 7 [7:5]
*                REFREQ_reg_val - the value for RFREQ to be writtent to LO registers 8 [4:0], 9, 10, 11, and 12
*                                 (this will need to be converted from a double to the actual reg value)
*/
void Write_LO_Values(int N1_reg_val, int HS_DIV_reg_val, uint32_t RFREQ_reg_upper_val, uint32_t RFREQ_reg_lower_val);
```

```
/*
 * N1_Lookup
 *  - Description:  Helper function to convert the regular number used in the frequency generation math
 *                  into the value that actually needs to be written to reg 7 [4:0] and reg 8 [7:6]
 *
 *                  Legal values are 1 and multiples of two.  Illegal odd values are rouned up.
 *                  The value written to the register should be the desired divider minus one.  Ex: if you
 *                  wanted N1 = 10, you would write 0b000_1001 (9 in decimal).
 *
 *  - Inputs:       N1_number - the regular number used in the frequency calculations
 *  - Outputs:      N1_reg_val - the value to be written to the LO registers corresponding to the value used in the math
 */
uint32_t N1_Lookup(int N1_number);

/*
 * HS_DIV_Lookup
 *  - Description:  Helper function to convert the regular number used in the frequency generation math
 *                  into the value that actually needs to be written to reg 7 [7:5]
 *
 *  - Inputs:       HS_DIV_number - the regular number used in the frequency calculations
 *  - Outputs:      HS_DIV_reg_val - the value to be written to the LO registers corresponding to the value used in the math
 */
uint32_t HS_DIV_Lookup(int HS_DIV_number);

 /*
 * RFREQ_Lower_Lookup
```

```
 *  - Description:  Helper function to convert the decimal portion of the floating point RFREQ value into the
 *                  lower half of the value that will be written to the LO registers
 *
 *  - Inputs:       RFREQ_number - the floating point number used in the frequency calculations
 *  - Outputs:      sum - the sum of all of the decimal point values after they've been multiplied
 *                                                  by 2^28
 *
 *  NOTES:
 *    As of 3/19/21, there is error introduced into the calculation (the lowest 3 hex characters of RFREQ are inaccurate)
 *    However, after recalculating the frequency using the new RFREQ value w/error introduced, it doesn't seem like it will
 *    have an appreciable effect on the result.  Hopefully this is good enough.
 *
 */
uint32_t RFREQ_Lower_Lookup(double RFREQ_number);


/*
 * RFREQ_Upper_Lookup
 *  - Description:  Helper function to convert the integer portion of the floating point RFREQ value into the
 *                  upper half of the value that will be written to the LO registers
 *
 *  - Inputs:       RFREQ_number - the floating point number used in the frequency calculations
 *  - Outputs:      RFREQ_upper_reg_val - the integer portion of RFREQ to be written to the upper register
 */
uint32_t RFREQ_Upper_Lookup(double RFREQ_number);
#endif
```