EE2033: Mini Project Report

Group: B02-11

Wenrui Wu A0177885E
Tianyu He A0177829J

Introduction

For this mini-project, we are supposed to complete two tasks.

- Task 1

  For task 1, we are required to design a low pass filter to filter out the interference and thus get a message with less noise. It is expected that a higher PSR (packet success rate) can be obtained using a short period of time. While designing the filter, we are also expected to limit the order of the filter and the amount of op-amps used to minimize the cost.

- Task 2

  For task 2, we are required to modify the given grc file to transmit two messages with two different symbol rates simultaneously using OOK demodulation. We are also required to implement two switches: one is to swap the transmission messages, the other one is to choose which received message string to decode.

Task 1

1. Objective

In task 1, we are required to employ Adalm Pluto, RTL-SDR, Analog Discovery 2, Filter Pro, LTSpice, GNURadio with given specification (Design A), and design a LPF (low pass filter) to suppress the interference and recover the message signal. The objective is to maximize the PSR (packet success rate) while minimizing the time taken to achieve desired number of packets. On top of these, cost is taken into consideration.

Project Design Specifications	Unit	Design A
Pluto Sink Settings		
Samp_rate Pluto Carrier Frequency (fc) Symbol Rate (fsym) Signal Amplitude (Asig) Interference Frequency (fint) Interference Amplitude (Aint) Interference Symbol Rate	(Hz) (GHz) (Bits / sec) (Hz) (Bits / sec)	1.6M 0.7 160k 0.15 320k 0.3 40k
RTL-SDR Source Settings		
Samp_rate RTL-SDR Carrier Frequency (freq)	(Hz) (GHz)	1.6M 0.7
Analog Discovery 2 Settings		
Samp_rate	(Hz)	4M
Opamp_files Settings		
fs_in fs_out	(Hz) (Hz)	4M 1.6M

Figure 1: Specification Design A

2. Filter Design

Given that RTL-SDR has an intermediate frequency of 1.5 MHz, the message signal and interference will be shifted to center at 1.5MHz and 1.82MHz, respectively. To verify this assumption, we ran Waveform-Spectrum to give an analysis.

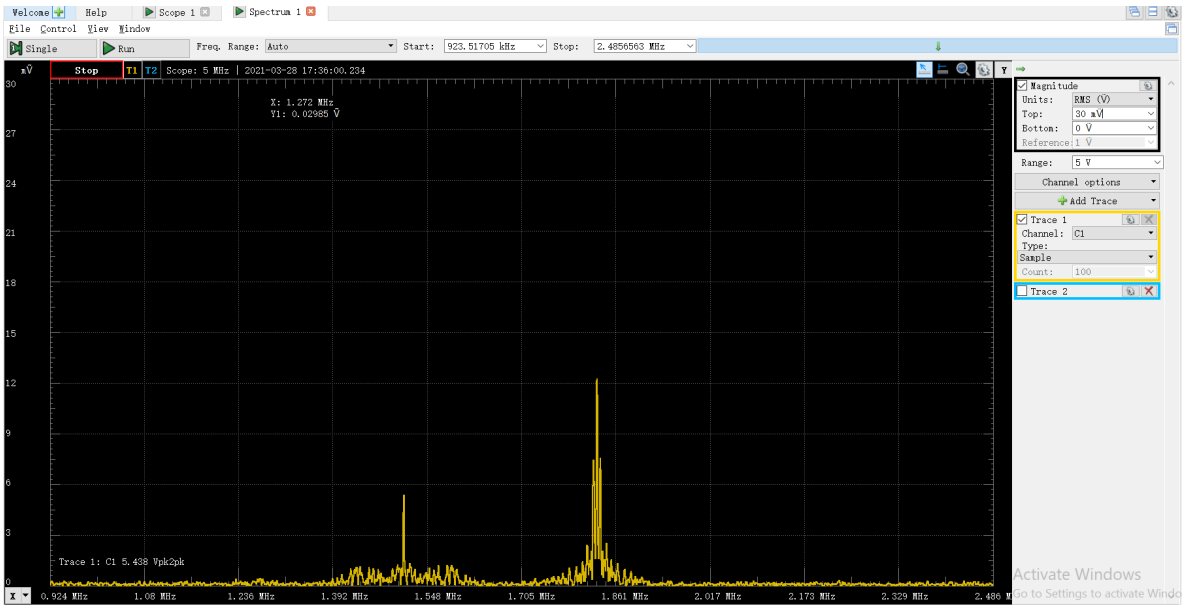


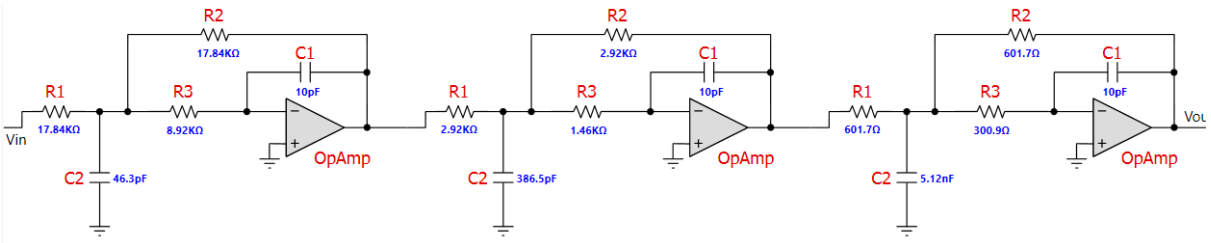
Figure 2: Spectrum Analysis

From the spectrum analysis, we found that the actual message signal is centered at 1.5073 MHz, while the noise signal is centered at 1.8294 MHz. The differences between actual frequency spectrum and design specifications are negligible. Noted the interference magnitude is about twice of message signal magnitude, which copes with design specifications.

Given that the first null bandwidth of message signal and interference are 160 kHz and 40 kHz, respectively, we managed to design a filter with cut-off frequency of 1.66Mhz and stop-band frequency of 1.78 MHz with attenuation of 10dB.

In Filter Pro, after setting up above parameters, only 7-order Chebyshev 1dB and 8-order Chebyshev 0.5dB design were provided. Here, we consulted GA about the order of filter, the suggestion was not to exceed 6-order. We also consulted peers, they suggested that we could use a Chebyshev filter since the interference is relatively close to the message signal in design A, which required a sharper attenuation. Keeping these in mind, we finally managed to change the stop-band

attenuation to be -8 dB while keeping all other parameters unchanged. The Filter-Pro design is shown below.



Filter Stage:	1	Filter Stage:	2	Filter Stage:	3
Passband Gain(Ao) :	1	Passband Gain(Ao) :	1	Passband Gain(Ao) :	1
Cutoff Frequency(fn):	586.2101 kHz	Cutoff Frequency(fn):	1.2397 MHz	Cutoff Frequency(fn):	1.6523 MHz
QualityFactor (Q):	0.761	QualityFactor (Q):	2.198	QualityFactor (Q):	8.004
Filter Response:	Chebyshev1dB	Filter Response:	Chebyshev1dB	Filter Response:	Chebyshev1dB
Circuit Topology:	MultipleFeedback	Circuit Topology:	MultipleFeedback	Circuit Topology:	MultipleFeedback
Min GBW reqd.:	44.6106 MHz	Min GBW reqd.:	272.4857 MHz	Min GBW reqd.:	1322.4928 MHz

Figure 3: Filter Pro Design Version 1

However, in LTSpice this filter design will cause an over-suppression at cut-off frequency, so we over-designed the filter. The final version of our filter design is a 6-order Chebyshev 1 dB filter with 1.7 MHz as cut-off frequency. The final version of Filter Pro design and parameters are shown below.

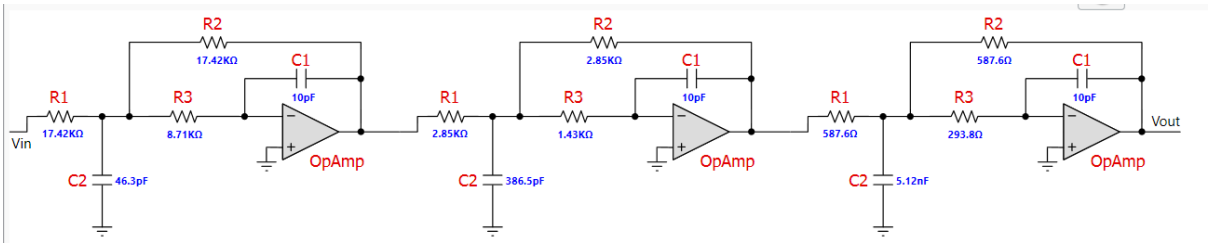


Figure 4: Filter Pro Design Final Version

Filter Stage:	1	Filter Stage:	2	Filter Stage:	3
Passband Gain(Ao) :	1	Passband Gain(Ao) :	1	Passband Gain(Ao) :	1
Cutoff Frequency(fn):	600.3357 kHz	Cutoff Frequency(fn):	1.2696 MHz	Cutoff Frequency(fn):	1.6921 MHz
QualityFactor (Q):	0.761	QualityFactor (Q):	2.198	QualityFactor (Q):	8.004
Filter Response:	Chebyshev1dB	Filter Response:	Chebyshev1dB	Filter Response:	Chebyshev1dB
Circuit Topology:	MultipleFeedback	Circuit Topology:	MultipleFeedback	Circuit Topology:	MultipleFeedback
Min GBW reqd.:	45.6855 MHz	Min GBW reqd.:	279.0516 MHz	Min GBW reqd.:	1354.3601 MHz

Figure 5: Filter Pro Design Final Version Parameters

We chose LM6172 (GBW: 100 MHz) for stage 1 and 2 and LM7171B (200 MHz) for stage 3. On top of meeting GBW requirements, the considerations include:

1. The GBW requirement for stage 2 is 279 MHz, using LM6172 will bring only a 2.79% error.
2. Due to our experiences, LM6172 performs better than LM7171B in actual design.

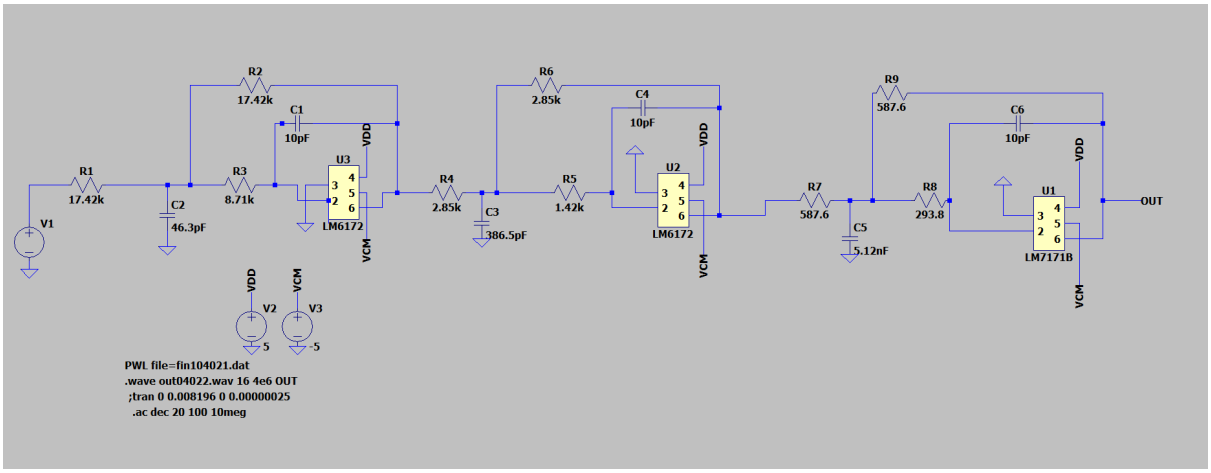


Figure 6: LTSpice schematic

The bode plot of the filter in LTSpiceis shown below. The gain at pass-band frequency is 0 dB, at cut-off frequency is -2 dB, and at stop-band frequency is -12 dB.

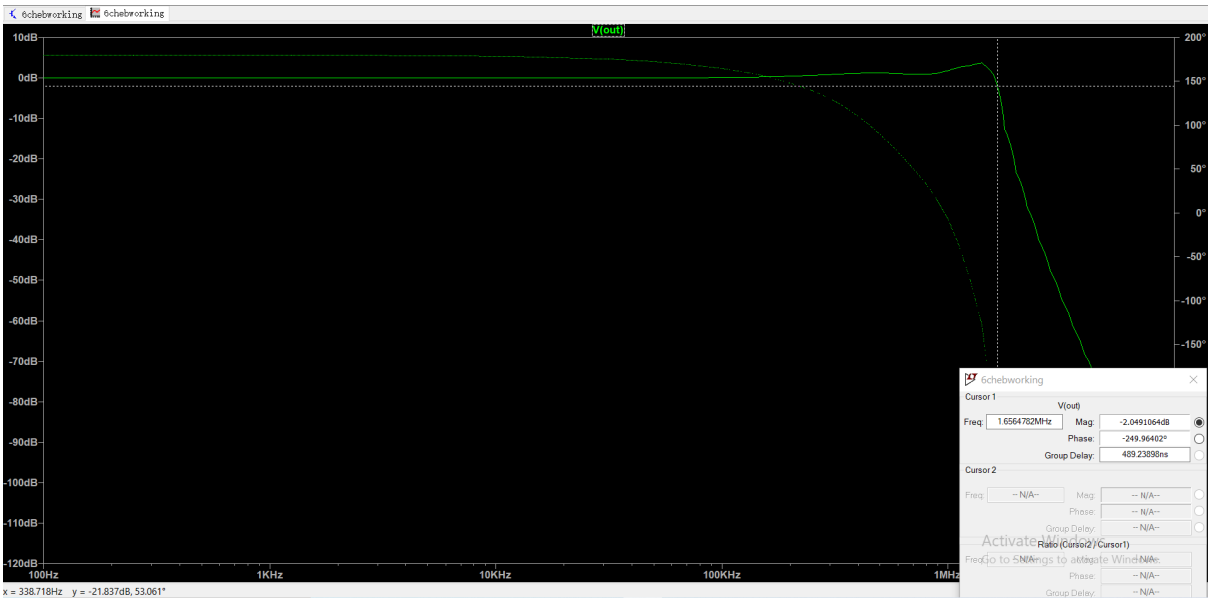


Figure 7: LTSpice Bode Plot

3. Implementation on the breadboard

Then we built the filter on the breadboard. In total, 3 op-amps (LM6172 x 2, LM7171B x 1), 9 variable resistors and 8 static capacitors were used. Although the stage 2 and stage 3 op-amps did not meet the min GBW requirement, the error is still acceptable.

After building the circuit strictly following the parameters given in Filter Pro, we found that the performance of the actual filter is unsatisfactory. According to the bode plot, the suppression at cut-off frequency is excessive while the one at stop-band frequency is insufficient. Here we managed to directly change the components values of our breadboard circuit.

In lecture 3, we learnt that the gain of each stage is given by

$$H_o = -\frac{R_2}{R_1}$$

and the cut-off frequency is given by

$$\omega_0 = \frac{1}{\sqrt{R_2 R_3 C_5 C_6}}$$

Also, from previous experiences that Capacitor has a relatively hard-to-control effect on the filter. We finally managed to change the R3 in each stage to meet the suppression. We used Waveform-Network to analyze the filter and changed value of R3 simultaneously. On top of these, we used two capacitors of 10 uF to connect V++ and V- -to ground respectively, in order to reduce large interference brought by power source and stabilize the system.

With these changes, our actual filter is shown below:

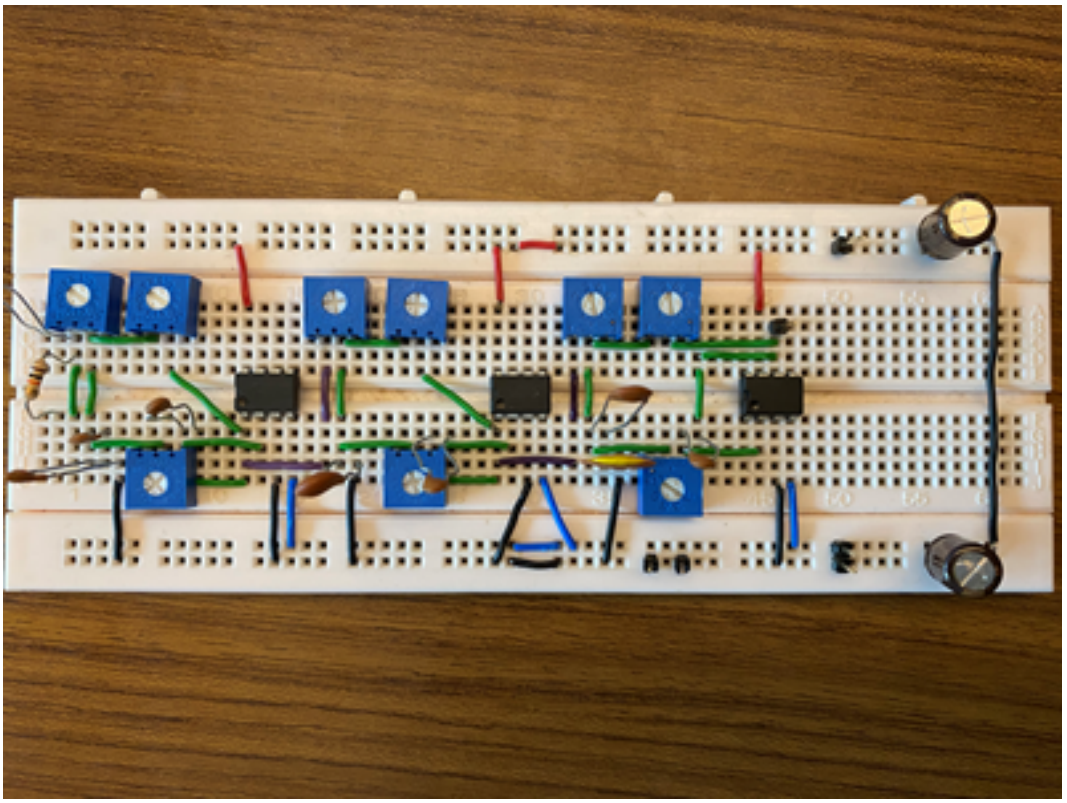


Figure 8: Actual LPF on breadboard

Compenent	Value	Unit
Stage 1		
R1 R2 R3	12.94 13.21 7.11	(kΩ)
C1 C2	10 46.3	(pF)

Stage 2		
R1 R2 R3	1.727 1.612 0.949	(kΩ)
C1 C2	10 391	(pF)
Stage 3		
R1 R2 R3	509 460 280.9	(Ω)
C1 C2	10 5.2	(pF) (nF)

Figure 9: Actual Filter Specifications

The Bode Plot of actual filter is show below:

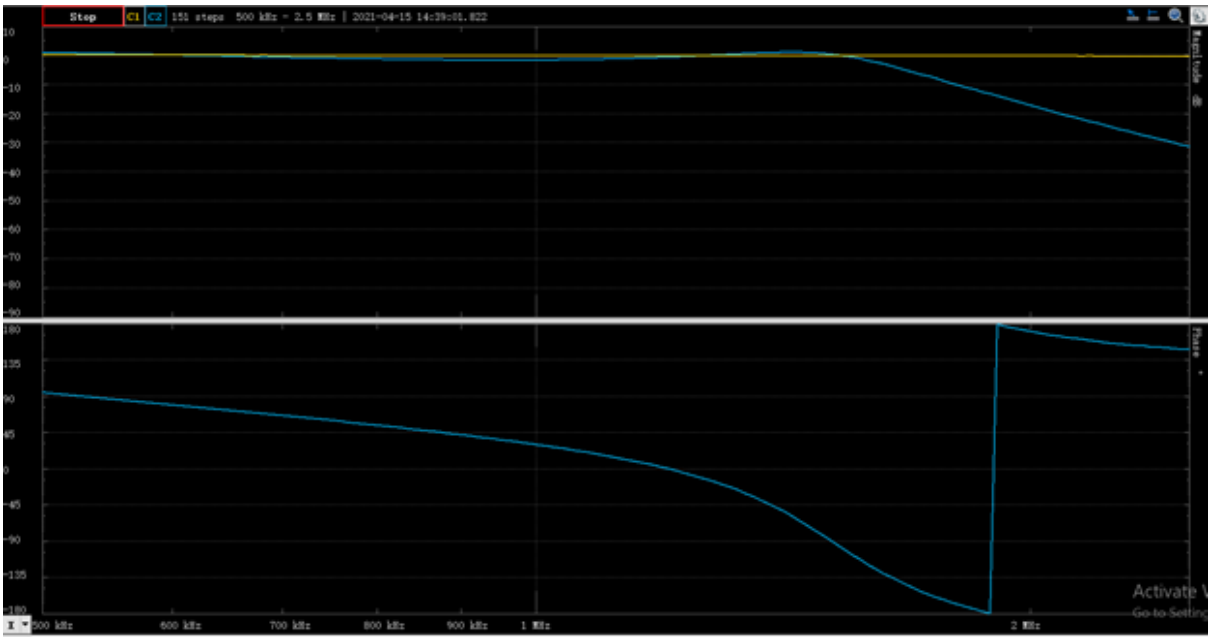


Figure 10: Actual Filter Bode Plot

4. Demonstration Results Summary

Our final demonstration results including PSR at 1000 packet and time taken to send are recorded as follows:

Interference	PSR	Time (sec)
1/10 Aint	0.579	2.746
1/3 Aint	0.597	2.857
Aint	0.212	3.101

Figure 11: Summary of results

5. Limitations & Solutions

1. During the filter design process, the relatively huge difference of Bode Plot between Filter Pro Design and LTSpice simulation, and between LTSpice simulation and Actual Filter actively demonstrates the fact that both Filter Pro and LTSpice can only give an optimal observation.
2. The length of wires and feet of capacitors can bring a huge interference which affects the performance a lot. When implementing the filter on the breadboard, we tried to minimize the length as much as possible.
3. To reduce the effect of power source V++ and V--, we connect two capacitors between power source and ground.
4. Different antenna impacts the performance of transmission quite much, that the longer antenna was chosen at last for it performed better.

Task 2:

1. Design idea

To be consistent with our design specifications given in task 1, we choose 1.6M as our sample rate. We assign the given symbol rates 160k and 80k to the two messages: Covid(msg_str1, 160k) and Cured(msg_str2, 80k).

Our idea is to use two carrier frequencies which are one lower frequency and one higher frequency to shift the two signals to the right to avoid interference. Then, we will transmit the shifted signals through the pluto sink. After the signals are received by the pluto source, we will use one low pass filter to isolate the signal located at the lower frequency and one high pass filter to isolate the signal located at the higher frequency. In order to implement the function of choosing which text message to decode, we designed four paths. Below is the figure showing the blocks of our grc file.

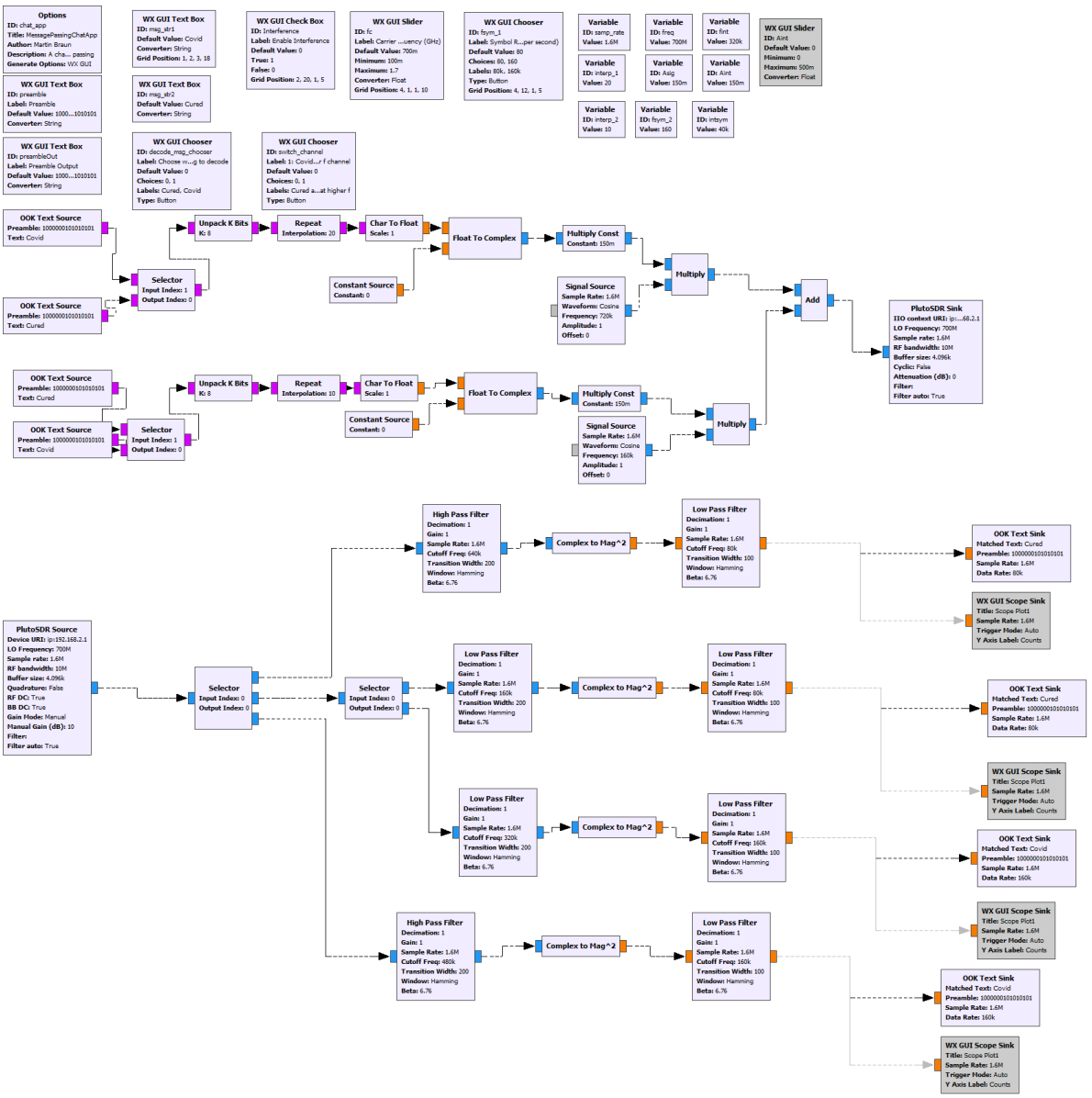


Figure 12: grc blocks

1.1 Transmitter side

As the first null bandwidth of the signal equals to the data rate, which is the symbol rate in our situation, the first null bandwidths for two messages are 160k and 80k when they are at the baseband. Once they are shifted to the right, the first null bandwidths will thus become 320k and 160k. As the half of the sampling rate is 800k, we can place one signal near to the baseband and one signal near to the 800k. Besides, as we may swap the transmitting channel for two messages, we design the carrier frequency to be dynamic to make sure the two signals are separated as far as possible. For example, if the “Covid” message with 160k symbol rate is transmitted at the higher frequency range and the “Cured” message with 80k symbol rate is

transmitted at the lower frequency range, the higher carrier frequency will become 640k, which is 800k-160k, and the lower carrier frequency will become 80k.

1.2 Receiver side

In order to implement the two required switches, we designed two WX GUI Chooser, which are decode_msg_chooser for choosing which text to decode and switch_channel for swapping the transmitting channel. Below is the table showing the meaning of these two choosers.

	Value = 1	Value = 0
decode_msg_chooser	choose to decode msg_str1: “Covid”	choose to decode msg_str2: “Cured”
switch channel	transmit msg_str1 in higher frequency channel, msg_str2 in lower frequency channel	transmit msg_str1 in lower frequency channel, msg_str2 in higher frequency channel

Figure 13: Meaning of WX GUI Choosers

At the first selector in the receiver side, we used the sum of decode_msg_chooser and switch_channel as the path chooser.

When the sum equals 2, this indicates that the msg_str1 is transmitted in a higher frequency channel and we choose to decode msg_str1. So we use a high pass filter with cut-off frequency at 480k, which is 800k - 320k, to filter out the useful signal.

When the sum equals 0, this indicates that the msg_str2 is transmitted in a higher frequency channel and we choose to decode msg_str2. So we use a high pass filter with cut-off frequency at 640k, which is 800k - 160k, to filter out the useful signal.

When the sum equals 1, there are two situations. One is that the msg_str1 is transmitted in a lower frequency channel and we choose to decode msg_str1. The other one is that the msg_str2 is transmitted in a lower frequency channel and we choose to decode msg_str2.

Here, we implemented another selector whose path chooser value is decode_msg_chooser. Thus, when decode_msg_chooser equals to 1, the first situation is employed. Therefore, we could apply a low pass filter with cut-off

frequency at 320k to filter out the msg_str1 signal. Else, when decode_msg_chooser equals 0, the second situation is employed. Therefore, we could apply a low pass filter with cut-off frequency at 160k to filter out the msg_str2 signal.

After filtering out the corresponding signal and thus shifting the signal back to the baseband, we implement another low pass filter using the symbol rate of the decoded message as cut-off frequency to further filter out excessive noise. Then, we further decode the corresponding text using ook text sink.

2. Results

By changing the corresponding WX GUI Chooser value inside the grc file, we can successfully decode the choosing message with a PSR around 0.6 and swap the transmitting channel for each message. However, when trying to change the transmitting channel or decode the message using the button inside the pop-up interface, the interface will become not responding. One solution to this problem could be changing the selector method to the multiplying constant method.

Conclusion

In conclusion, our team managed to design, simulate, build, and tune a 6-order Chebyshev 1dB low pass filter to suppress the interference located at 1.82MHz while preserving the original signal. In addition, we also modify the given grc file to try to transmit two messages with different symbol rates at the same time and decode them using the OOK modulation and demodulation.

What we have learned from task 1 is that simulation is an essential step for filter design. By observing the bode plot in LTSpice, we can notice the difference between this bode plot and the bode plot obtained from the Filterpro. However, it is more important to adjust our filter design based on the real bode plot of the built circuit. Besides, the tuning process should be carried out using the theory taught in the class. By observing the bode plot as well as the spectrum, we can tune the resistor and capacitor values respectively. In task 2, we also learned the meaning of each grc block and gained a deeper understanding on the OOK modulation and demodulation. To sum up, this project helps us to gain a better understanding of real-life filter design.