18-747 WDD Lab 2: Modeling a Radio System

Reese Grimsley, Department of Electrical & Computer Engineering, Carnegie Mellon University
reeseg@cmu.edu
October 11, 2019

***Abstract*—This report is concerned with learning GNURadio, characterizing wireless channels and modulation schemes, and establishing an understanding of the link budget. GNURadio is an open source software for modeling radios and communication networks, with the ability to program Universal Software Radio Peripherals (USRPs) such that real radio signals can be captured and processed without highly specialized hardware. Data collection performed in this lab provides useful information for learning how noise, attenuation, and reflections affect a radio systems ability to receive and decode signals adversely affected by the environment.**

I. Introduction

Wireless communications have been a massive boon to technology as electronics become ubiquitous and pervasive. From a distant perspective, the fact that it works at all should baffle most people as invisible waves transmit information without any physical connection. From a much closer perspective, the fact that wireless communication works is still baffling as one becomes familiar with all of the difficulties involved with transmitting electromagnetic waves that carry information such that said information can be gleaned from the bruised and battered wave that arrives at the receiver. As we discussed in class, leaving the wire is a near death sentence for signals, so we need intelligent techniques to protect those signals from noise, interference, reflections, etc. To develop these techniques, it is important that we know the enemy. By understanding the nature of these dangerous conditions, we can take steps to avoid them, be it through different modulation techniques, filter designs, frequency band selection, etc.

This lab serves as a great compliment to a course I took previously, which focused on the more theoretical side of digital communications, covering topics from source encoding to modulation to channel encoding and the corresponding reversals. The theory of digital communications is an extremely rich area that pulls heavily from signal processing and statistics. We often assign probabilistic properties to the aggressors of the wireless channel, *i.e.,* noise, interference, and multipath. Having this background has made this lab considerably more interesting as my previous course was mainly concerned with hard, theoretical limits, such that I have a decent understanding of SNR and its effect on error, the Shannon Limit, error correction, and encoding principles (both source and channel). I find this background helps drive some intuition on what is happening when we attempt to model these channels in more practical scenarios.

In this report, I describe the procedure and corresponding results for using GNURadio, an open source application for simulating and synthesizing communications equipment, for several different tasks intended to characterize the wireless channel under non-ideal conditions. The primary findings of these experiments will relate how conditions of the wireless channel affect our ability to decode radio signals; these decoding methods are largely considered a black box, but knowledge of the inner workings are especially helpful in drawing conclusions from the findings of this lab.

II. Related Work

As briefly discussed in the Introduction, digital communications is an extremely rich field that has revolutionized the way society shares information. Much of this theory is not necessarily specific to wireless communication, but is rather generalized to information transfer in general. A central figure of this field of information theory is Claude Shannon, an American scientist whose name graces the EE sacred stones of the Shannon-Nyquist Sampling criterion, which states that a digital signal can accurately capture all information in an analog signal so long as it is sampled at twice the highest frequency component of that signal, and the Shannon-Hartley Channel Capacity, shown in Equation 1, which defines the maximum bitrate over a channel for bandwidth *W* and signal to noise ratio *SNR*. In a previous life, I was able to reproduce the proof for the Shannon Coding limit, which defines the minimum number of bits required to encode a symbol within a symbol alphabet based on the probability distribution of symbols (more specifically, symbol entropy) [1].

$$C = Wlog_2(1 + SNR) \tag{1}$$

Since we are interested in communicating digital information through a wireless channel, we need not only digital communication theory but also electromagnetic and radio propagation topics. While distance has an effect on signal strength through a wire, the effect is nowhere near as fatal as the $P \propto 1/R^2$ trailoff that occurs due to the fact that the transmitted power is spread around a sphere originating from the transmitting antenna. Simple geometry describes this relation of power and distance, but the real case is worse as we also consider the frequency of our signal also affects the effective receive power. Equation 2 shows the Free Space Path Loss (FSPL) relation between transmit and receive power for a transmitted wave of frequency f Hz with distance d m of separation. For simplicity, this assumes ideal, isotropic antennas with directivity of 1. This relation to frequency is a result of the relationship between effective area of an antenna

and the wavelength of the incident EM wave. I note that our next lab is on antennas, so I will let further discussion rest until then.

$$Pr/Pt = (\frac{c}{4\pi df})^2 \quad (2)$$

Unfortunately, the above relations that describe transmit power loss are only a part of the battle. I mention that wireless channels kills signals, and this is tragically not a peaceful death of old age that our electromagnetic friends would experience from free space path loss alone. FSPL tells us what happens at opposite ends of the channel, but nothing of what happens between. Here, the transmitted signal experiences noise, interference, and reflections that lead to delayed copies of the signal that can jumble up the receiver if not handled carefully. The traditional model for noise in communication systems is additive white gaussian noise; this is because thermal noise is approximately white, meaning that it is a constant value in its frequency domain form. As a fun fact, true white noise is a physically unrealizable construct, as it would imply a signal of infinite energy (Johnson-Nyquist thermal noise trails off around 10^12 Hz [2], so effectively white for our purposes). This noise power can be represented as $P = kBT$, which at room temperature comes out to around -150 dB with respect to 1 W of power for a bandwidth *B* of 200 kHz. Much of this noise actually occurs at the receiver, and is a function of temperature more than the channel itself. Another large reason for error is interference, in which other signals superimpose with the desired one. If the interferer is out of band, it can be ignored using effective analog filters in the front-end of the receiver; however, we have no such option if it is within the band we are interested in. This is why multiple access control (MAC) protocols are so important in wireless networks. These define what should happen if collisions, *i.e.,* two transmitters talking at the same time, occur and prevent signals from being decoded. Note that these only apply to devices on the same protocol. There are fewer anti-collision options if the interfering signals are using different protocols!

The last foe we must consider is reflections and multipath. When an EM wave is reflected by an object, it potentially finds another (longer) path to the receiver. If this path is only a little bit longer *i.e.,* on the order of one wavelength, then we have a time delayed signal that superimposes with the transmitted signal that took the more direct path. Clearly, there can be multiple reflections. Reflections may also occur from distant objects; these lead to what we call inter-symbol interference, where the difference in path lengths is on the order of many wavelengths. In this scenario, the delayed signal is so much further behind that it actually contains a previous symbol.

III. Approach

In this section, I delineate the process for each part of the lab procedure. In summary, part 1's purpose is to build an FM receiver in GNURadio, part 2's purpose is to measure symbol error rate for varied transmit power, near reflections and far reflections for two modulation constellations, and part 3's purpose is to find the extent to which SNR limits a LoRa receiver's ability to decode signals when the noise floor or attenuation of the transmitted signal is increased. This third part measures the link budget of the receiver and channel.

A. FM Receiver

This part is a standard "paint by numbers" process, meaning that the directions are all laid out clearly; there is little imagination required for creating the FM receiver in this portion of the lab.

For sake of completeness, I describe those steps here

1) Create a new GRC file with the WX GUI
2) Change the sample rate variable to 250k
3) Create new variables for audio interpolation (set to 4) and audio rate (set to 44.1k)
4) Add a WX GUI slider labelled 'freq' and enter default value of 94.1 M
   a) Set minimum value to 88.1 M, maximum to 107.9 M, and num_steps to 99
5) Add another slider labelled 'gain', with a default of 70, min of 0, max of 90, and 100 steps
6) Create a "UHD: USRP source" for interfacing with the USRP
   a) In the RF options tab of this block, set CH0: center freq as 'freq', Gain as 'gain', and Antenna as 'TX/RX'
7) Create a Rational Resampler with interpolation as audio_rate*audio_interp
8) Create a WBFM Receive with Quadrature rate = audio_rate*audio_interp, and audio decimaiton = audio_interp
9) Create an audio sink with sample rate = audio_rate
10) Connect the blocks

When this is run in GNURadio, a window appears that shows an FFT of the incoming signal, with the center frequency set by the Frequency slider. By cycling through the frequencies, one can hear different radio stations through the computers speakers. Further information about the outcome of this task is located in the following section. I do not believe there are assumptions worth stating for this part of the lab given its simplicity.

B. The Wireless Channel

For this part of the lab, the goal is to characterize the wireless channel by measuring error rate while varying different types of noise in the channel. The noisy channel is simulated by adding white Gaussian noise, nearby reflections, and distant reflections. Nearby reflections are considered to be close enough that phase offset of the transmitted and reflected signal at the receiver is less than a full carrier wave period, *i.e.,* less than 360 degrees. Distant reflections are considered to be on the order of many cycles, *i.e.,* offsets up to one symbol period. These types of noise are viewed independently, such that no more than one noise source is affecting the channel at a time. In addition to these types of noise, two modulation schemes, or constellations, are used to see how the wireless channel differently affects binary phase shift keying (BPSK) vs. quadrature amplitude modulation (QAM). Only

two bits are used for the QAM symbols, making this QAM-4 modulation synonymous to quadrature phase shift keying (QPSK).

The following steps outline the concrete steps followed to collect data for this part of the lab. Transmit is abbreviated TX, symbol error rate is SER, and signal to noise ratio is SNR. For each of the items in this section, I collect 10 samples of data per parameter tuple, which will be used to report average and standard deviation in the following section.

1) First, turn TX power to 0 and noise to maximum. Take note of the constellation and SER.
2) Keep TX power at 0 and reduce noise to 0 dB. Look at the FFT plot.
   a) Take note of the power level.
   b) Increase TX power to maximum, and note the SNR.
3) Incrementally raise noise by 10 dB and note SNR at each noise value. Be sure to capture the SNR that first gives non-zero SER
4) Set noise back to 0, and begin varying parameters related to nearby reflectors.
   a) Specifically, measure SER for different reflection amplitudes (increment by 0.1 between 0.5 and 1) at different phase offsets (increment by 45 degrees between 0 and 360).
   b) Observe the time-domain and constellation plots to see how these change due to local reflection.
5) Set nearby reflection parameters back to 0, and begin varying parameters related to distant reflectors.
   a) Specifically, measure SER at different reflection amplitudes (increment by 0.1 between 0.5 and 1) at different symbol offsets (increment by 0.1 between 0 and 1).
   b) Observe the time-domain and constellation plots to see how these change due to distant reflection.
6) Up to this point, 4-QAM has been used. Change the constellation variable to PSK, and set the number of bits variable to 1. Recollect data for 3-5 using Binary PSK.

For my case, I found that the averaging window of 1000 samples was not enough, and the delay rate was faster than I preferred. To change this, I modified two blocks. I changed the error detection block to use a 20k sample window, and the SER display block (QT Gui Number Sink) to use an update rate of 2 seconds.

1) Assumptions:

- First and foremost, I assume that the GNURadio flow graph for this portion of the lab is correct.
- The averaging window is not sliding; each new average is on a totally new set of values compared to the previous average
- Demodulation always happens the exact same way
  - This is motivated by the fact that the constellation in the display window appears to change over time
  - *i.e.,* demodulation is not an adaptive process in GNURadio

C. Link Budget

This final part of the lab is an investigation into the link budget, which is a term to account for how much noise and interference we tolerate without losing the ability to decode packets. The link budget is often understood as a value in decibels describing the acceptable SNR that will allow correct decoding of a packet. This is affected by receiver sensitivity, transmit power, antenna directivity, and the wireless channel. In this portion of the lab, we primarily look at transmit power (varied by attenuating the transmitted signal) and the wireless channel. The wireless channel is modified by adding noise; we are not considering the effect of multipath here. LoRa is the modulation scheme used for this portion of the lab, so the message length and spreading factor are also modified to analyze their effect on the link budget. There is a LoPy module setup that sends LoRa signals through an attenuated wire connection to the USRP, which decodes the signals and shows raw data in the display.

The following steps specify the procedure followed for data collection in this part of the experiment. When changes are made to any LoRa parameter, the GNURadio display needs to be closed and reopened (but not the entire program).

1) First, view the display with default settings. The waterfall diagram should show transmissions, and the logs at the bottom of the main GNURadio should show what message was sent, along with the TX power and spreading factor (SF).
2) Looking at the frequency plot, use the max hold feature to identify the transmission. Note the SNR between this max value and the noise level.
3) Modify the TX power from 6 to 20 in increments of 2. Note the SNR and the ability to decode packets; is there any change in this?
4) Reset TX power to 20. Increment the noise in 10dB steps, noting SNR at each step. Take note of when packets are no longer decoded.
5) Reset noise to 0. Increment signal attenutation in 10dB steps, noting SNR at each step. Take note of when packets are no longer decoded.
6) Reset attenuation to 0. Modify spreading factor between 7, 8, 9, and 10, and redo steps 4 and 5.
7) Modify the message lengths for spreading factors of 7 and 10. For this part, I use messages of length 3, 15 (the default), and 50 (the max). Redo steps 4 and 5 for each of these messages and note how this changes the decodability for different SNR.

It is important to note that the message length pertains to the message used in the variable block. The true message length, based on the source code, appears to have substantial overhead due to the frame header and LoRa metadata that is also sent.

1) Assumptions:

- Similar to the previous part, I assume the GNURadio program is correct.
- SNR should be measured as the difference between max peaks of the signal and noise.
- The message sent by the LoPy includes TX power and spreading factor.

  - This increases overhead by at least 35 bytes.
- When the FFT plot displays value of power in dB, this is with reference to 1 W.

IV. Experimental Results

This section is dedicated to providing quantitative results from this lab. Since the first part (building an FM receiver) requires a primarily qualitative analysis, this portion is left to the following section. Instead, data and plots are provided only for parts 2 and 3 since these required much more intensive numerical data collection. For each of these subsections, my goal is to linearly follow the lab directions to aid clarity (at the expense of the organic flow that a more typical academic paper might have). Assume that, unless otherwise specified, that all dB values are with reference to 1 W

A. The Wireless Channel

The first task in this section is to set zero transmission power and maximum noise. An IQ plot of this is shown in Figure 1. As one might expect, the received points in the IQ plane resemble a random scatter plot. The GNURadio block generates this as white gaussian noise. The mean value of SER falls around 0.75; this is to be expected because even a random choice will be correct for 1/4 of the samples, given that there are 4 options for 4-QAM.

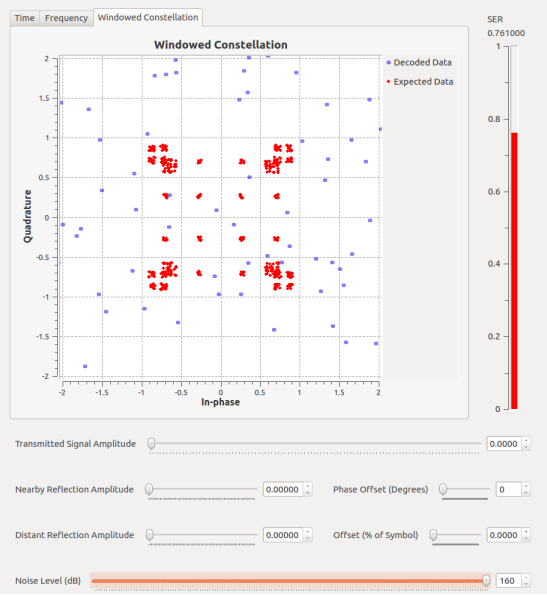

Fig. 1: Pure Noise Constellation

The second task has us changing the noise level at the receiver. First, we are to look at the noise floor; Figure 2 shows this to rest at around -160 dB. Recall from section II that we stated the thermal noise to be -150 dB at room temperature for a 200kHz wide band. My supposition is that this noise floor is derived from thermal noise, although it is unclear how this is modeled since this part of the lab is performed in software only. This establishes our baseline noise. As we increase the noise, we would expect the SER to increase; this is reflected in Figure 3 as we see that noise increases the SER, but appears

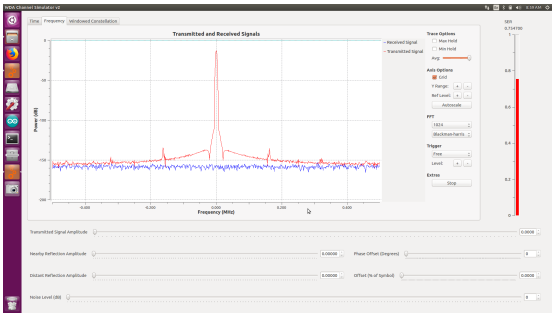

Fig. 2: Receive power with no transmitted signal

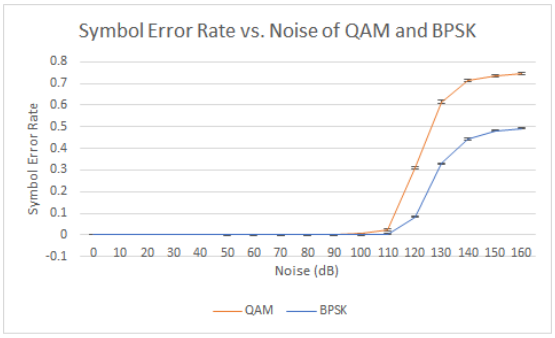

Fig. 3: Symbol Error Rate for varied additive white Gaussian noise

to have little effect until it is around 100 dB. The first errors begin to show at SNR of 90 dB and 73 dB for QAM and BPSK, respectively.

The third task is intended to show the effect of reflectors in the Fresnel zone, such that those reflections add less than one wavelength to the total path traveled by the reflected wave. From the receivers perspective, this manifests as a phase offset that is superimposed with the unperturbed transmission. The results for QAM and BPSK are shown in Figures 4 and 5. Clearly the angle of reflection and modulation scheme have a large effect on the error rate. More information is given in the following analysis section.

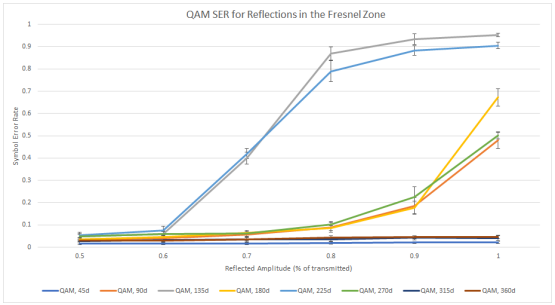

Fig. 4: QAM Symbol Error Rate for local reflections

The fourth task is closely related to the third: we are again interested in the effect of reflections on the error rate, but now the reflection delays are on the order of a symbol period. Refer to Figures 6 and 7 for QAM and BPSK error rates, respectively, for varied reflection power at various symbol offsets.

It is worth noting that the error, shown with error bars based on the standard deviation resulting from 10 samples,

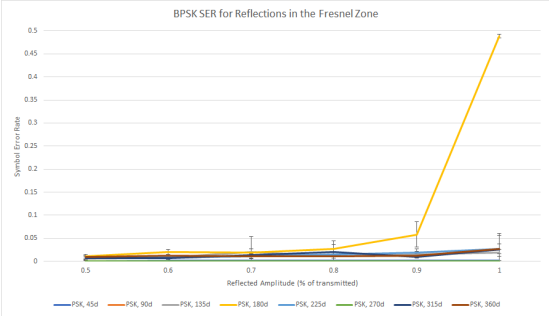

Fig. 5: BPSK Symbol Error Rate for local reflections

is significant. I found that there was much variation in the SER, even for a large averaging window of 20k samples. Unfortunately, some of these error bars overlap, making it difficult to discern the exact amount of error for each reflection offset.

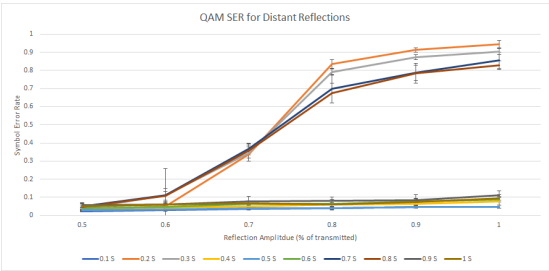

Fig. 6: QAM Symbol Error Rate for distant reflections

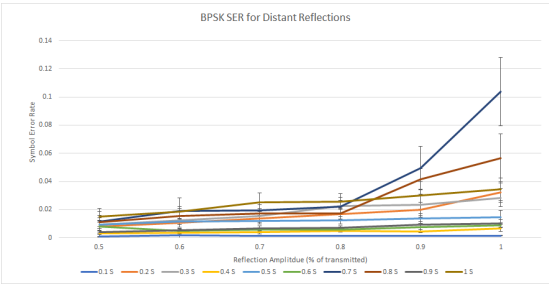

Fig. 7: BPSK Symbol Error Rate for distant reflections

B. The Link Budget

The very first task is simply to see the transmission within the waterfall plot and ascertain that the messages are being decoded in the console. Figure 8 shows this waterfall plot, in which the short, yellow lines indicate a transmission. As soon as one of these shows up, the console displays the transmitted message, "*(26714) Message: 'Welcome to WDA!' @ power 20 and SF7*." I note that just above this are many hexidecimal values that correspond to the ASCII values of all but the first 3 characters in this transmission. This means that the typed message to send is not the only thing sent: metadata is sent over LoRa as well, confirming my assumption about overhead in the transmission.

The second task is to use the frequency plot to find the power level of the transmission at default parameters, *i.e.,* TX

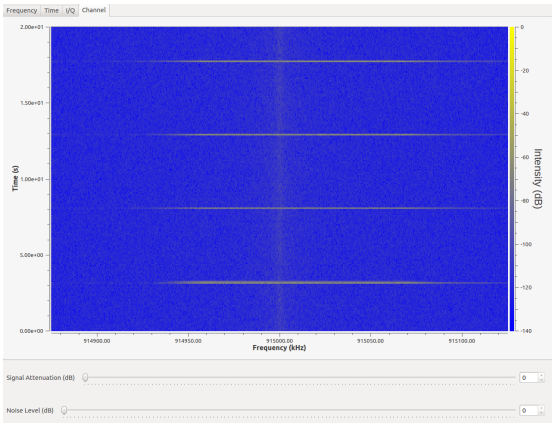

Fig. 8: Waterfall graph of LoRa transmission with default parameters

power 20 and spreading factor 7. I set the averaging slider low to find the peak average noise value of -105 dB at the center of the channel, and used max-hold to find a value of -25 dB for transmission power; I estimate the SNR at 70 dB. Based on my assumptions, the absolute power levels fo noise and RX power are in dB with respect to 1 W.

The third task looks at the effect of the TX power setting for a LoRa radio. Specifically, we look at the changes in RX power or SNR, which will shown identical trends, assuming that the amount of noise does not substantially change while taking these measurements. See Figure 9; as one might expect, increasing the TX power linearly increases the SNR because RX power increases, but noise is unchanged.

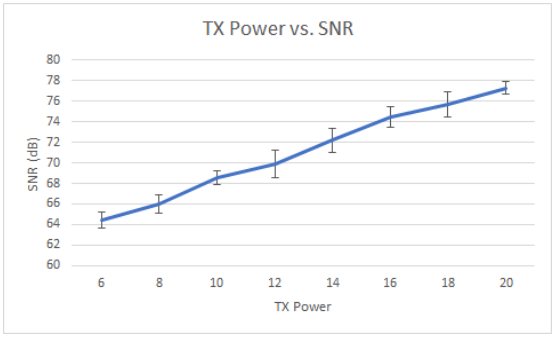

Fig. 9: Transmission power's effect on Signal to Noise Ratio

In the fourth task, we look to the SNR when noise is varied. There are several later tasks that require more data to be collected under different spreading factors and message lengths for the sake of measuring the SNR at which transmissions are no longer decoded properly. However, the SNR does not change for these because spreading factor lengthens the time per bit and message length changes the number of bits; both of these affect time and energy consumed, but not power itself. For this reason, I only show the SNR values for various noise levels for the default LoRa values of spreading factor 7 and message length 15 bytes (not counting overhead and metadata bytes). Figure 10 shows this data. I notice that messages stop being decoded when noise is around 66 dB (SNR at 23 dB)

The fifth task is extremely similar to the previous one, but instead of increasing the noise, we go the opposite direction

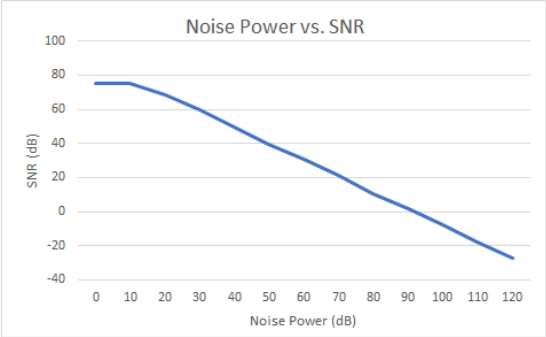

Fig. 10: Noise vs. Signal to Noise Ratio

TABLE I: SNR of Last Decoded Packet due to Noise

Spreading Factor	Message Length (bytes)	SNR (dB)
7	15	23
8	15	23
9	15	22
10	15	22
7	3	22
10	3	22
7	50	22
10	50	22

and attenuate the transmitted signal. Just like the previous task, this set of data collection is redone at different message lengths and spreading factors, but the SNR vs. attenuation trends are the same among them; all that changes is the ability to decode the message. Refer to Figure 11 for the relation between SNR and attenuation at SF 7 and message length of 15 bytes; I notice that messages stop being decoded when attenuation is 42 dB (SNR of 34 dB - this is 11 dB higher than when noise was the independent variable!).

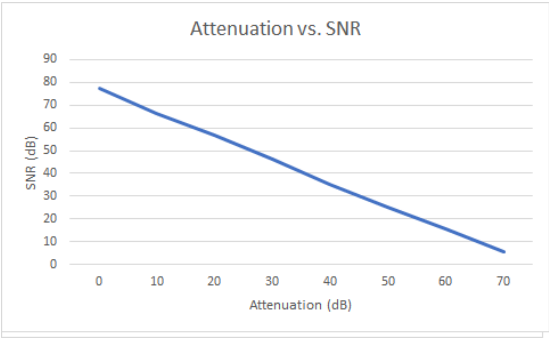

Fig. 11: Attenuation vs. Signal to Noise Ratio

The last two tasks ask for the noise and attenuation to be varied once again, but for different values of the spreading factor and message length. For sake of reducing this report's length, I am reporting the lowest SNR that the message could be decoded at for these spreading factor, message length pairs, but I do not include plots of the attenuation vs. SNR or noise vs. SNR because this is a rehash of Figures 10 and 11. See Table I for lowest decodable SNR values for varied noise and Table II

TABLE II: SNR of Last Decoded Packet due to Attenuation

Spreading Factor	Message Length (bytes)	SNR (dB)
7	15	34
8	15	31
9	15	28
10	15	25
7	3	34
10	3	25
7	50	34
10	50	25

In addition, I was able to capture the time-domain signal of a chirp; without knowing the frequency domain characteristcs of this signal, one would be hard pressed to make any sense of this jumble. Please see Figure 12.

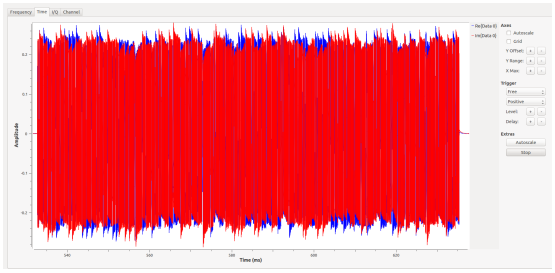

Fig. 12: Time-Domain Chirp

V. Analysis

Looking back at data from the previous section, I make several observations about each of the tasks.

Creating an FM receiver in GNURadio was much more straightforward than expected. However, it becomes immediately clear when looking at other GNURadio flow graphs how much more complicated these can be made much more complex by including various settings, diagnostics, plots, etc. My expectation had originally been that cranking up the gain would linearly amplify noise and audio (if present) in the channel. This turned out to be mostly wrong; in fact, increasing the gain drowned out some of the noise in channels that were not picked up well at lower gains.

The second task has a much meatier analysis compared to the first part since there was substantially more data to collect.

The first part more or less establishes an upper bound on the symbol error rate at $(n-1)/n$ where n is the number of symbol possibilities. I will note that this is not technically driven by the number of symbols, but the probability distribution between symbols. However, since we generate uniformly distributed random numbers, $(n-1)/n$ is the correct relation for maximum SER. What I found interesting with the randomness is that the constellation points transmitted are not singular values; they are spread out over an area of IQ values whose distribution I cannot explain (and not for lack of trying). I cannot find any parameters in the flow graph that appear to shape the distribution besides *Differential Encoding* in the modulator, but this does not have the effect that I was

looking to explain. In particular, I have noticed that leaving the module running longer actually causes some of these red dots in Figure 1 to change in ways that reflect some of the noise when said noise is more patterned. I found these patterns to change primarily when collecting data on multipath.

The second part of this task on the wireless channel is more or less straightforward; as the noise increases, SER increases. At around 110 dB, this becomes more significant and tops out at $(n-1)/n$ near 140 dB. QAM and BPSK follow a similar trend here, with QAM increasing slightly faster, even accounting for the maximum error bound.

The third and fourth parts of analyzing the wireless channel deal with reflections, leading to phase and symbol offsets. As the phase or symbol offset increases, there is not a linear increase in SER; in fact, there is little similarity between the same type of offset change and the SER for different modulation schemes. The constellations help explain this, as the reflections visibly shift the received points into different parts of the IQ plot, some of which are particularly damaging to SER. I found that these offsets effectively rotated the RX signals about the expected points; for phase offset, there was a single rotation made wihin 360 degrees, but there were two full rotations made for symbol offsets between 0 and 1 symbol. Due to the simplicity of BPSK, I notice that there are far fewer phase and symbol offsets that lead to an intolerable error rate. This is because BPSK has such a simple decision boundary, *i.e.,* the Q-axis. Since both I and Q values matter for QAM, changes to these are substantially more damaging. It should not surprise us that packing more bits into a symbol makes decoding more difficult! In the case of multipath due to near or far reflections, this appears to be substantially worse for 4QAM than BPSK. In general, it is not difficult to see patterns in the offsets that cause greater error. For instance, phase offsets between 225 and 315 degrees are substantially more damaging than those closer to 0 on the unit circle. Symbol offset gives similar information, but is slightly more difficult to generalize. In general, it seems that symbol offsets are more dangerous than phase offsets.

The third task's main point of analysis, in my opinion, is the SNR required to decode packets for different spreading factors and message lengths. It is fairly obvious from Tables I and II that the message length has little effect on the decodability. This is contrary to my expectation, in which a longer message would be more difficult to decode because there is less room for error. However, longer messages allow longer error checking codes; this offers a partial explanation. An interesting point to note is that there is far more variation in lowest usable SNR for attenuated signals than noisy signals. Between SF7 and SF10, there is a 9dB difference in SNR that the message can be decoded at when the signal was attenuated, but only a 1dB difference when the signal was kept at max power but noise was added. Let us note that the overall minimum SNR for noisy signals is lower than the required SNR for attenuated signals despite seeing less variation. A possible explanation is that the raw power of the received signal must be above some threshold that is dependent on spreading factor for the message to be decoded at all. For example, received signals at SF 7 with information content below -70 dB (w.r.t. 1 W of power) cannot be decoded effectively, even though the SNR may be good enough to otherwise decode the signal. With this explanation, it would make sense that we can increase the noise to cause lower SNR but still receive the signal because the raw power of the information-carrying signal is plenty strong for the receiver's hardware. Nonetheless, this still runs orthogonal to my supposition that a higher spreading factor would make packets easier to decode in general since the time per bit (or chip, to be technical) is greater for higher SF.

VI. Conclusion

For this lab, I performed experiments to measure the wireless channel and the link budget using GNURadio. I looked at the various elements that make the wireless channel a dangerous place to operate; the symbol error rate has a direct depedence on noise and multipath effects, and these dependence is also affected by the way information is encoded into the symbols. By measuring the signal to noise ratio between the received power and transmit power, I have characterized elements of the link budget, which tells us the tolerable SNR differential that still permits reliable decoding of packets. With this knowledge in mind, it is now much easier to grasp how we must carefully design our wireless networks. If nothing else, this develops a deeper appreciation for the engineers who have integrated wireless connectivity so seamlessly into our daily lives!

References

[1] A. Kolmogorov, "On the shannon theory of information transmission in the case of continuous signals," *IRE Transactions on Information Theory*, vol. 2, no. 4, pp. 102–108, 1956.

[2] C. S. Turner, "Johnson-nyquist noise," *url: http://www. claysturner. com/dsp/Johnson-NyquistNoise*, 2012.