Analysis of Filtering and Noise Threshold of Wireless Signals

Jacob Hollenbeck, *Member, IEEE,* and Jake Groat, *Member, IEEE*

***Abstract*—This paper examines the effect of white noise on an IEEE 802.11b wireless signal. We establish the basics of the 802.11 data structure and transmission signal before investigating the raw transmission and reception characteristics. Our experimentation involves finding a threshold at which the signal becomes unusable. Multiple filters are employed and compared to determine which configuration delivers the most dependable information.**

***Keywords*—IEEE, 802.11b, white-noise, Lowpass filter.**

I. Introduction

Devices operating in the 2.4 GHz frequency are often subject to interference. This is due to the plethora of devices that emit signals at this range. Some examples may include microwave ovens, Bluetooth devices, baby monitors, and cell phones [4]. The objective of this project is to find the point at which the noise/interference makes a wireless signal unreadable. Simulated radio transmission signals will be generated with in the 802.11b protocol at 2.4GHz. These signals will be encrypted with direct-sequence spread spectrum (DSSS) modulation technique. Noise will then be injected into these signal to find the noise threshold for this signal. The noise threshold will be defined as the limit at which the signal can no longer be demodulated or is too mutated to be considered a quality data transmission.

Computer models for signal and data processing were used in Matlab/Simulink to simulate the transmitted and received signals. Simulink runs simulations of complex systems and Matlab primarily does mathematical operations.

II. Problem Description

Determining the maximum noise which can be absorbed and then filtered from a signal and still retain viable information is a constant challenge for signal processing engineers. Utilizing simulation and analysis tools to explore many methods of filtering enables the engineer to test filter solutions in a mathematical environment before sinking time and money into building physical systems at the risk of irreversible damage to components or equipment. In this paper, we test how various filters affect the noise threshold for a given signal. For the purposes of this test we will employ an IEEE 802.11b wireless signal.

Perhaps the biggest problem in wireless networking are signal collisions. Collisions occur when multiple clients are being received at the same time by a single server channel. 802.11 attempts to solve this problem by employing Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), called Distributed Coordination Function(DCF) as the basic channel access protocol. The standard also uses a polling scheme called Point Coordination Function (PCF) to grant stations access to the channel [5]. Reliable receipt and decoding of many stations is an essential function of wireless networking due to the large number of clients which may be present at any one time on the network. This makes the understanding of filtering and noise threshold even more important when designing and analyzing digital signals.

III. Data

The Matlab Communication System toolbox includes a model of the 802.11 WLAN protocol. The model builds a pseudo MAC header and Protocol Data Unit (MPDU). The test signal will consist of packets of 1's and 0's to simulate an outgoing data packet. We will use the 802.11b standard during simulation to transmit the data. The 802.11b uses direct-sequence spread spectrum (DSSS) modulation to transmit a given signal operating within the 2.4 GHz frequency. The 802.11b uses a 22 MHz bandwidth.

DSSS encodes the original signal using a sequence of pseudo random 1 and -1 values as seen in Figure 2. This pseudo random sequence is known as a Barker Code. A Barker code is a sequence of 1 and -1's that maintain a property of ideal auto correlation [1]. The encrypted signal is then phase-shifted with signal of much higher frequency. This spreads the signal across a vary large spectrum, using a large bandwidth. The result is a signal that looks similar to noise. This can be seen in Figure 3.

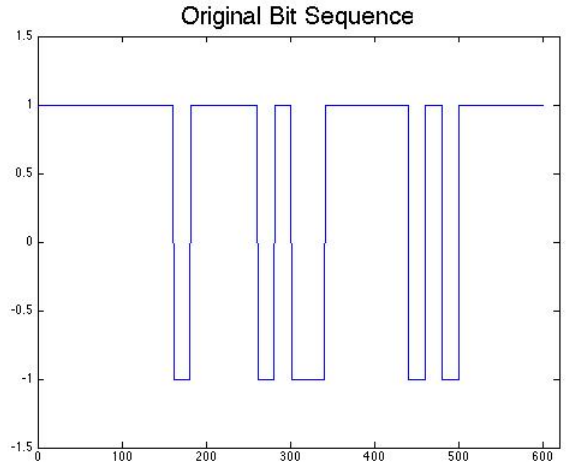

Fig. 1. DSSS Original Bit Sequence

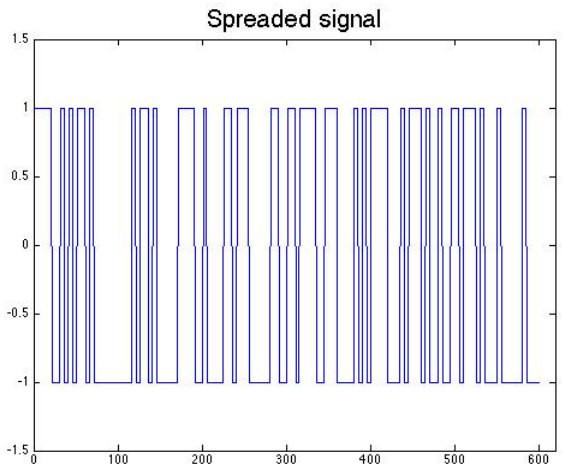

Fig. 2. DSSS Spread Signal

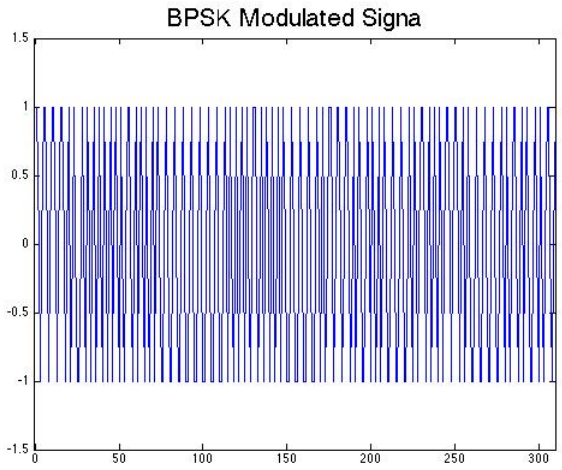

Fig. 3. DSSS Spread Signal

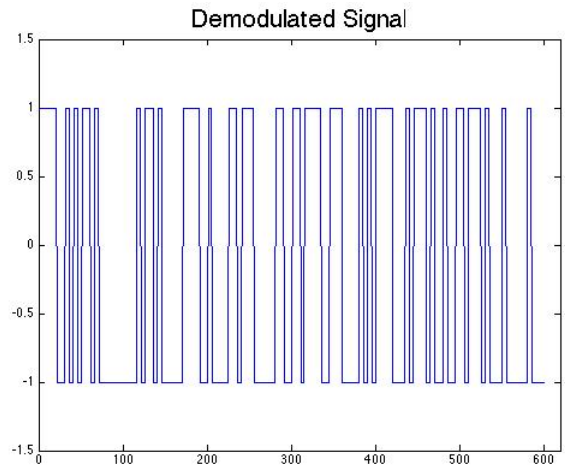

Fig. 4. DSSS Despread Signal

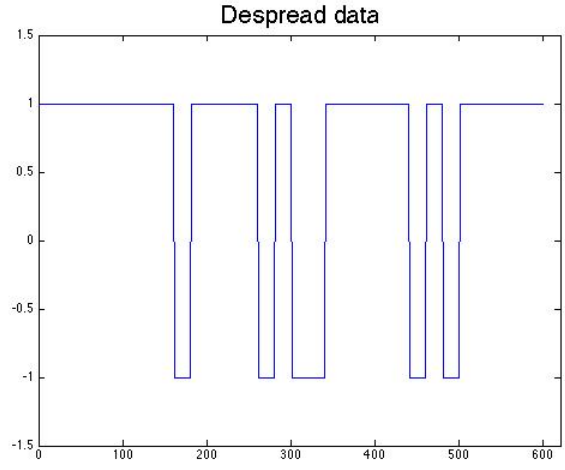

Fig. 5. DSSS Original Bit Sequence

The sequence used to encode the signal is now used to decode it. Because of the high correlation property of the barker code, the original signal can theoretically be perfectly recovered. This is done by multiplying the demodulated signal with the barker code for the duration of its bandwidth as seen in Figure 5.

A parameter used to determine the strength of the signal is the signal to noise ratio or SNR. The SNR is not actually a ratio but the difference in decibels between the received signal and the background noise [6]. The SNR is also referred to as the signal density. Signal strength is often quantified on routers using this metric.

Sample code that demonstrates the DSSS modulation code was found. The original source for this code can be found on mathworks [2]. This code was manipulated to output a series of plots demonstrating the sequence of events that take place in DSSS modulation.

IV. Evaluation Criteria

Because 802.11b is a data packet dependent application, the test will use the successful reception and decoding of the MPDU data packet as the basis for our evaluation. The Simulink model outputs the MPDU error rate which is graphed. The 802.11b protocol throws an error flag, represented by a logical one, when the data packet is unusable. If there is no error, the flag is a logical zero and the MPDU is decoded. To visually display the MDPU error rate, we graphed the error as a function of time. Figure 6 shows 50 ms of reception where all of the MPDUs were successfully decoded. Note because the beacon is only transmitted periodically, there is an error during times of non-transmission. However, we see that for each period of transmission, the error is a logical zero, indicating proper reception and decoding of the MPDU. For our deciding threshold we will consider a greater than 50% error rate as a loss of signal.

Fig. 6. Succesful Reception and Decoding of MPDU

V. Proposed Solution

While a signal is in transition, it is apt to be affected by noise from an outside source. The application of a lowpass filter will remove the unwanted noise from the generated signal at the device receiver. This will be implemented in Simulink using the filter design block.

VI. Simulation and Analysis

The test model simulates an 802.11b beacon frame. This frame is periodically transmitted to maintain the connection between the base station and client stations. Figure 7 displays the Matlab/Simulnk model which was used for testing and analysis.

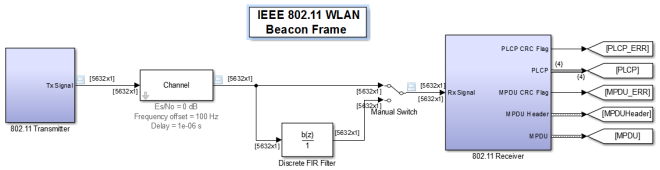

Fig. 7. Matlab Simulink 802.11b Model

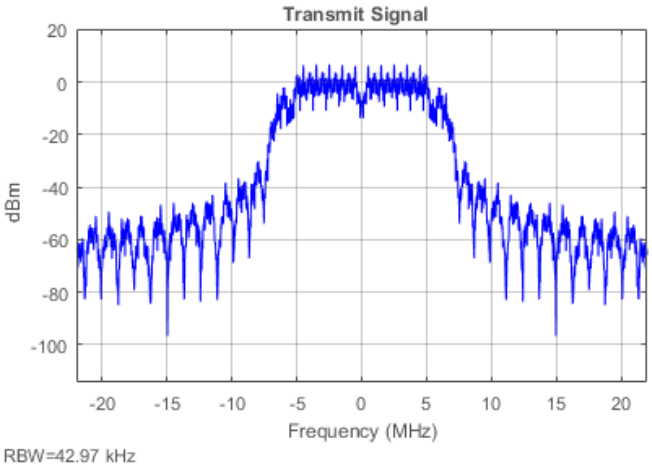

Fig. 8. Transmitted Signal

The raw transmitted signal, before the noise channel is shown in Figure 8. The received signal, with an SNR of 0db, which indicates the information signal has as much power as the noise, is shown in Figure 9. It's corresponding MPDU error rate is 37.5%, as shown in Figure 10.

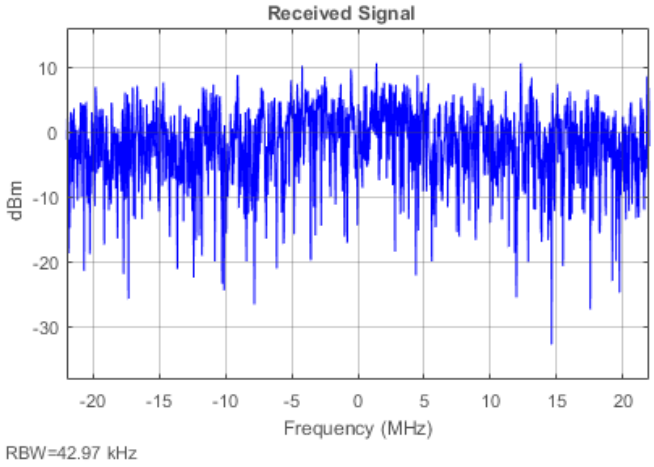

Fig. 9. Received Signal SNR=0dB

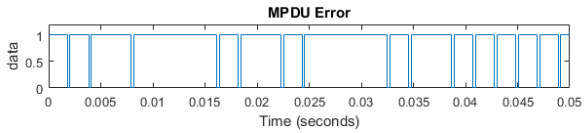

Fig. 10. MPDU Error SNR=0dB Error rate 37.5%

To test the effect of a filter before the receive section, we designed a 40th order filter using Matlab. The frequency and phase response of the filter is shown in Figure 11. The filtered receive signal is shown in Figure 12. The effect of the filter is clearly seen. However the error rate of the filtered signal does not improve when compared with the non-filtered signal. Figure 13 shows the error rate of the filtered signal to be 41.7% which is, in fact, a higher error rate than the non-filtered signal.

This section of the test was negative due to the difficulties in filtering white noise. To better test filtering, a known source of noise should be applied to the signal which can then be removed with a corresponding filter.

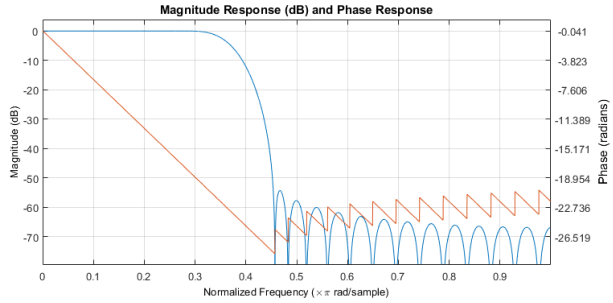

Fig. 11. FIR Filter Frequency and Phase Response

To find the noise threshold where 50% of the packets are lost, we started with a high signal to noise ratio of 5dB. We stepped through 5 different SNR levels to find the >50% packet loss.

As the signal level approaches an SNR of 0dB, we can see

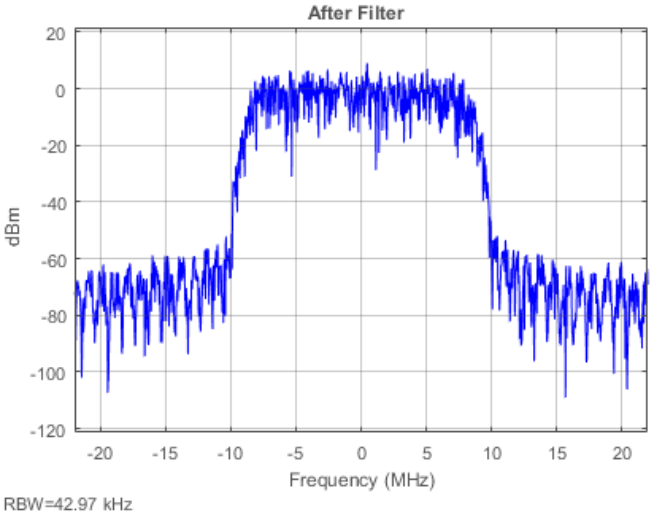

Fig. 12. Received Signal after FIR filter applied

Fig. 13. MPDU Error after Filtering, Error rate 41.7%

Fig. 14. MPDU Error SNR=5dB Error rate 0%

fewer and fewer MPDU packets being decoded, indicated by the missing zeros in the graph.

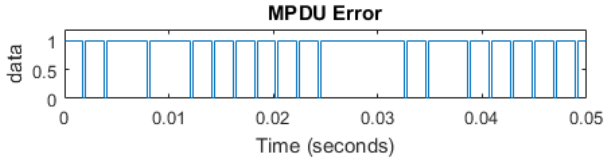

Fig. 15. MPDU Error SNR=1dB Error rate 25%

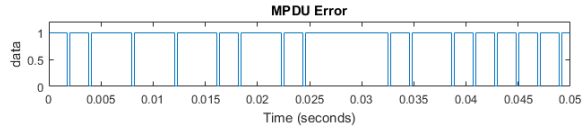

Fig. 16. MPDU Error SNR=0.5dB Error rate 33%

Finally at an SNR of -1dB, shown in Figure 18, we have lost 50% of the data packets. We can assume any signal weaker than this will introduce an error greater than the threshold of 50%.

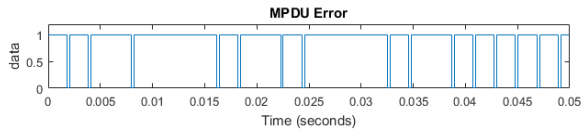

Fig. 17. MPDU Error SNR=-0.5dB Error rate 38%

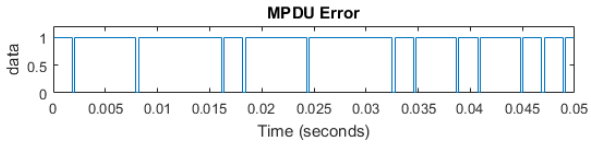

Fig. 18. MPDU Error SNR=-1dB Error rate 50%

VII. Development

To more effectively test the designed filter, future development would involve the application of different types of interference to the signal while it is in the channel. By injecting high frequency impulses, the effects of the filter would be better quantified. After receiving quantifiable results, the improvement from the application of different filters would become more discernible as well. Further testing would involve the effects these filters have on different modulation techniques. Modulation techniques used in other wireless standards would include Frequency-hopping spread spectrum, Orthogonal frequency-division multiplexing, and Multiple Input-Multiple Output Orthogonal frequency-division multiplexing. Due to time constraints these developments were not implemented.

VIII. Conclusion

This project was a great learning experience. To appear competent, we needed to learn about the different wireless signal protocols, the various modulation techniques, as well as different filter implementations. Tools such as Matlab and Simulink were also used. Unfortunately, we did not go as in depth as we should have into the different types of noise that is applied to signals. Ultimately, this led to a disproportionate amount of resources going into simulation. During simulation and analysis, it became clear that filtering the white noise, spread-spectrum signal was having little to no effect. By applying a filter to a signal that was spread out across such a broad spectrum, pieces of the signal were actually being cutoff. After further research, it was found that ”spread spectrum systems have a threshold or tolerance level of interference beyond which useful communication ceases” [3]. This means that it would take an impulse at a high frequency to significantly distort the signal. One important aspect that we failed to comprehend was the definition of white noise. White noise is a noise that is evenly distributed across all frequencies. The attempt to filter out this noise is beyond the scope of this class.

REFERENCES

[1] R. H. Barker. *Communication Theory*. 1953.

[2] Sanjeet Kumar. *Direct sequence Spread spectrum*. URL: http://www.mathworks.com/matlabcentral/fileexchange/24491-direct-sequence-spread-spectrum/content/Direct_sequence_spread_Spectrum.m.

[3] Prabakar Prabakarn. *Tutorial on Spread Spectrum Technology*. URL: http://www.eetimes.com/document.asp?doc_id=1271899.

[4] *THE MODEL OF COMMUNICATION CHANNEL IN THE 802.11B STANDARD WIRELESS NETWORK*. URL: https://dspace.vsb.cz/bitstream/handle/10084/83905/AEEE-2008-7-1-2-59-Nemec.pdf?sequence=1.

[5] Matthijs A Visser and Magda El Zarki. "Voice and data transmission over an 802.11 wireless network". In: *Personal, Indoor and Mobile Radio Communications, 1995. PIMRC'95. Wireless: Merging onto the Information Superhighway., Sixth IEEE International Symposium on*. Vol. 2. IEEE. 1995, pp. 648–652.

[6] *Wireless fundamentals: Signal-to-Noise Ratio (SNR) and wireless signal strength*. URL: https://documentation.meraki.com/MR/WiFi_Basics_and_Best_Practices/Wireless_fundamentals%3A_Signal-to-Noise_Ratio_%28SNR%29__and_wireless_signal_strength.