ECE 591: Software Defined Radio

Midterm Project - Baseband Digital Communication Transceiver Design

We certify that this work is original and not a product of anyone's work but our own.

Michael Bisbano

Jin Feng Lin

Submitted: Thursday, March 17, 2022

Due by: Thursday, March 17, 2022

Graded by: _____ Date: _____

Contents

List of Figures

List of Tables

List of Files

Abstract

The objective of the project is too simulate the digital baseband communication system. Specifically, the desire goal is to generate the transmitted baseband signal using different kinds of pulse shaping technologies. Then study the spectrum of the transmitted signal to investigate the frequency leak and bandwidth efficiency of different pulse shaping. Meanwhile, decoding the digital data set out from the transmitter with process of an ideal receiver.

1 Introduction

In the transmitter design, a user is allow to input the desire message from keyboard which the input will be translated into digital data using 7-bit ASCII code. These 0s and 1s will be the data (in bit): $b[i]$. Then, the digital data will be encoded into symbols ($S[i]$) based on the which baseband modulation scheme the user preferred. Using the encoded symbol sequence, a chosen pulse generator $p(t)$ is used to generated the transmitted baseband signal $y(t)$. The pulse generator $p(t)$ will generate a time-limited pulse, which will have to delay the pulse at time delay τ to modulate the symbols as shown in the equation below:

$$y(t) = \sum_i S[i] * p(t - iT - \tau) \tag{1}$$

The modulation types that will be applied to the signals includes on-off keying, 2-PAM, 4-PAM, 8-PAM, along with pulse shaping technologies such as rectangular, Hamming, Hanning, and Sinc. On-off keying, abbreviated as OOK, is the simplest form of amplitude-shift keying (ASK) modulation that represents digital data which contain two symbols, 0 and 1, such that the bits in the $b[i]$ are grouped in groups of 1 that for bits equal to 1 will be assigned as the symbol 1 otherwise the symbol 0 as shown in table 1. 2-PAM share very similar behavior as the OOK which the only difference in that the 0s bits in $b[i]$ get assigned as the symbol -1 rather than 0 as shown in table 2.

Bit Representation	Symbol
0	0
1	1

Table 1: OOK Bit to Symbol

Bit Representation	Symbol
0	-1
1	1

Table 2: 2-PAM Bit to Symbol

For 4-PAM and 8-PAM modulation type, reflected binary code (RBC), also know as reflected binary (RB) or Gray code, is used, RBC is an ordering of the binary numeral system such that two successive values differ in only one bit. For 4-PAM, there exist four symbols -3,-1,1,3 where the bits in $b[i]$ are grouped in 2 bits such that 00 is assigned as -3, 01 is assigned as -1, 11 as 1 and 10 as 3, as shown in table 3. For 8-PAM modulation type, it contains four additional symbols compare to 4-PAM which are -7,-5,5,7 and that the bits in $b[i]$ is grouped into 3 bits a set, which the bits to symbol assigned can be found in table 4.

Bit Representation	Symbol
00	-3
01	-1
11	1
10	3

Table 3: 4-PAM Bits to Symbol

Bit Representation	Symbol
000	-7
001	-5
011	-3
010	-1
011	1
111	3
101	5
100	7

Table 4: 8-PAM Bits to Symbol

All of the parameter for an ideal receiver can be obtained once the received signal in captured in time domain. The first part of the ideal receiver is the sampler, when rectangular, Hamming

or Hanning pulse shapes are used, the optimal sampling time is:

$$t_i = \tau + \delta + (i-1) * T + T/2 \tag{2}$$

The second part of the receiver is the quantizer, which can be divided into two steps. First, it need to compensate the transmission attenuation by performing the following operation:

$$r'[i] = r[i]/g \tag{3}$$

where $r[i]$ is the samples of the received signal and g is the receiver gain. The second step is to convert each and every sample to its closest possible value. For example, in 2-PAM, only +1 and -1 are the possible transmitted values. So, if one sample is 0.78, it is compared with +1 and -1 and determined that +1 is closer to 0.78 than -1, so this value is convert to the +1. Mathematically, this is corresponds to:

$$d[i] = \begin{cases} +1 & r'[i] > 0 \\ -1 & \text{otherwise} \end{cases} \tag{4}$$

After quantizer, a decoder is needed to undo what the encoder did at the transmission side of converting to bit(s) to symbols. That is, all the received symbols need to convert back to the data bits, 0s and 1s, which can be perform by applying the reverse function of the encoder. Finally, using the 0 and 1 sequence and grouped in 7 bits, each and every 7 bits can be converted back to one ASCII code.

2 Laboratory Experimental Results

2.1 Transmitter Design

Figures 1 to 4 demonstrate the time domain signal and frequency domain spectrum of a baseband transmission using 8-PAM with pulse shaping: rectangular, hamming, hanning and sinc respectively of the text “hello world” with a random initial delay before transmission.

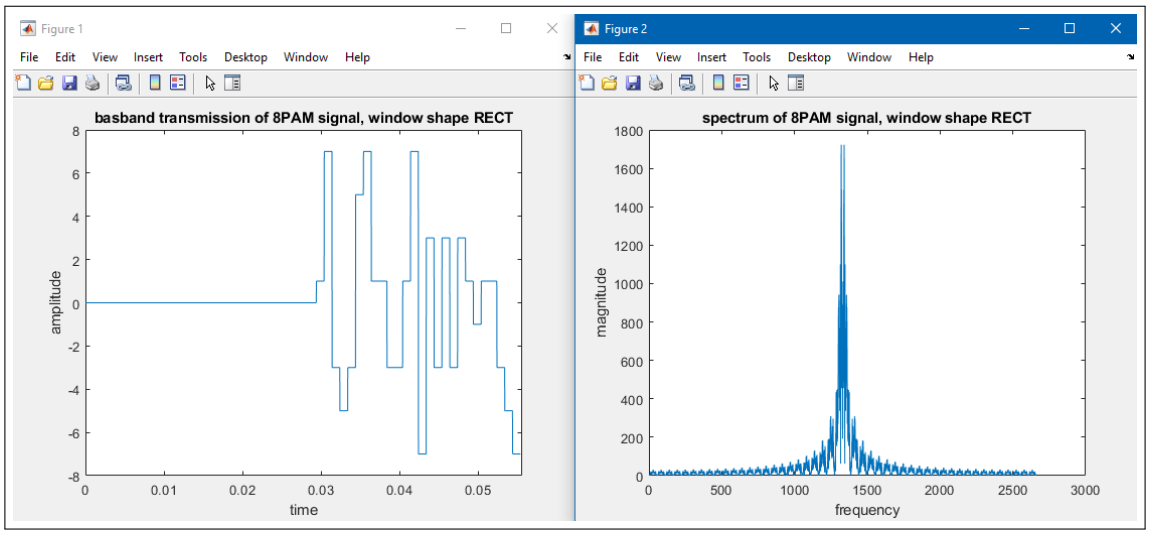

Figure 1: ”hello world” with 8-PAM with Rectangular in Time and Frequency Domain

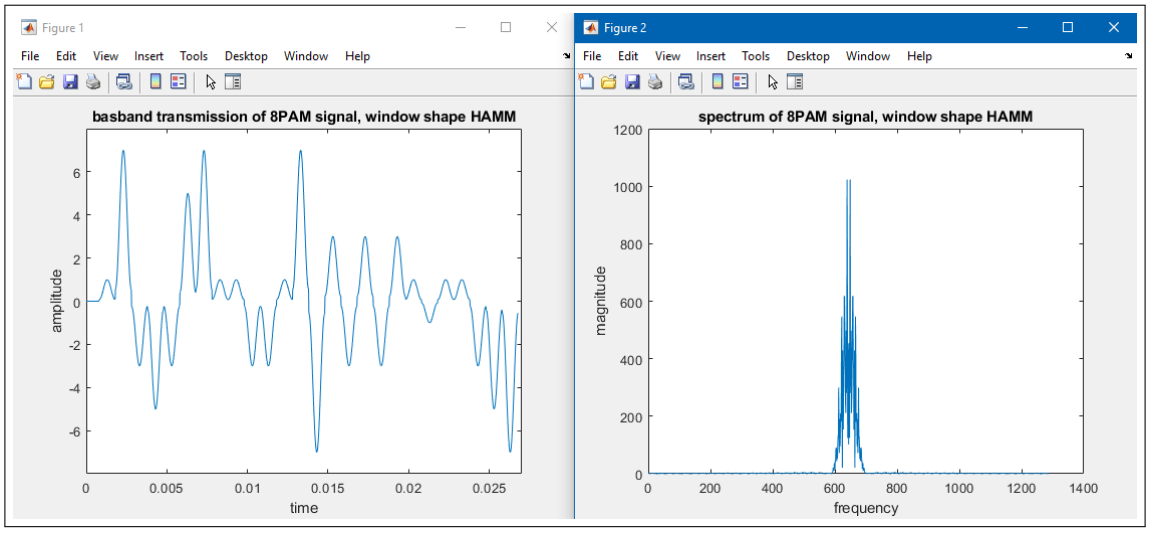

Figure 2: ”hello world” with 8-PAM with Hamming in Time and Frequency Domain

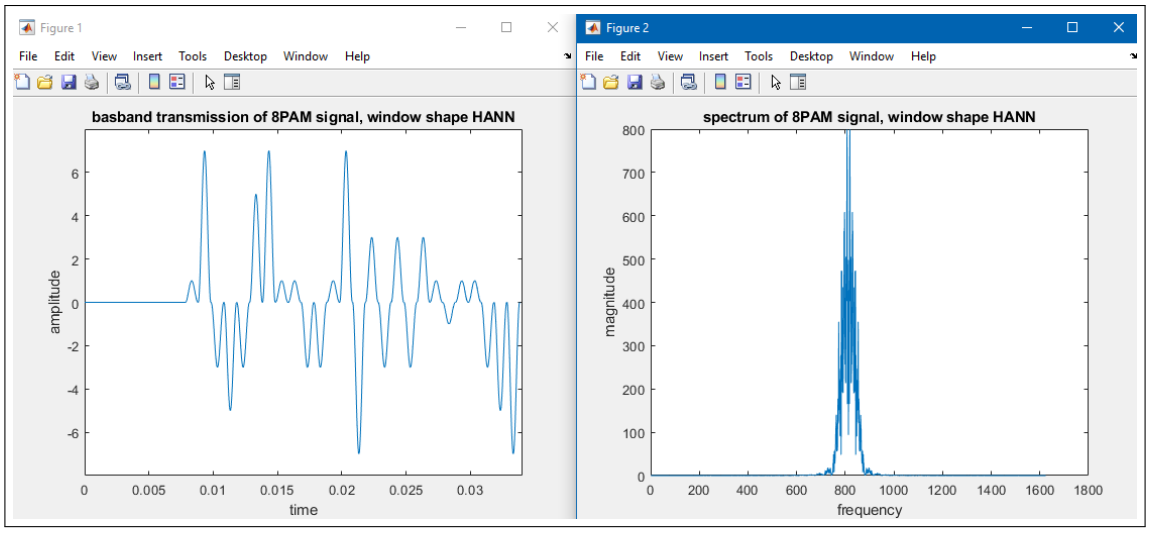

Figure 3: ”hello world” with 8-PAM with Hanning in Time and Frequency Domain

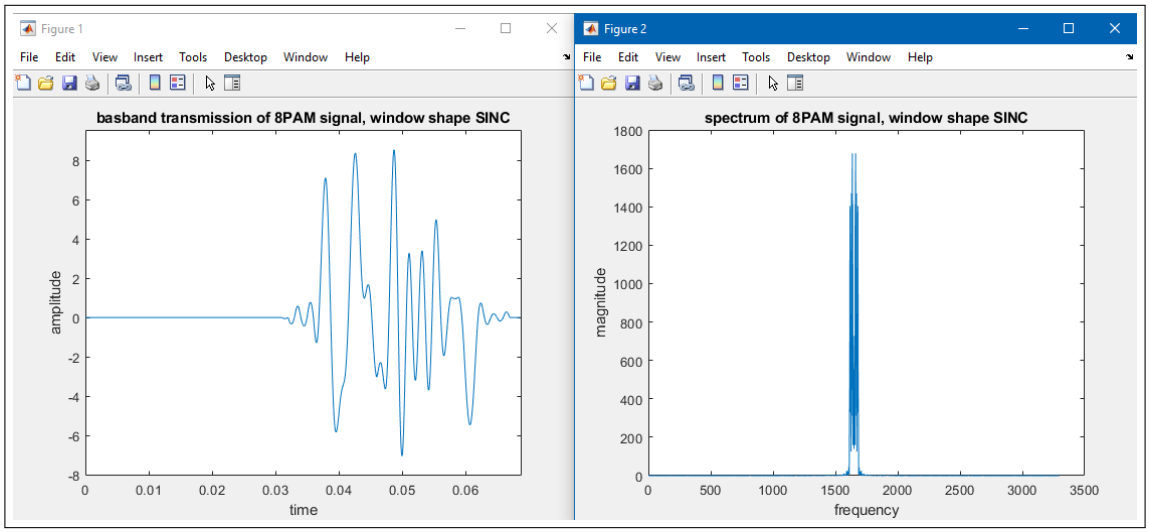

Figure 4: ”hello world” with 8-PAM with Sinc in Time and Frequency Domain

2.2 Receiver Design

Figure 5 to 8 represents the sampled data and the quantized data in the plot with decoded message “hello world!!!” display in the matlab command window for 8-PAM modulation type with pulse shaping of rectangular, hamming, hanning and sinc respectively.

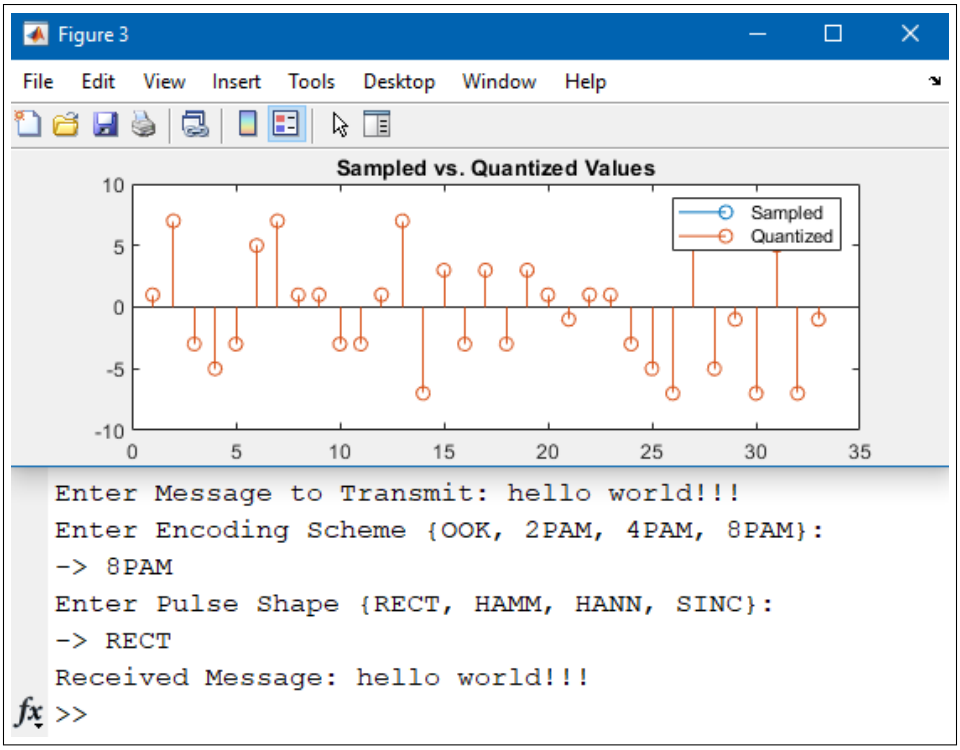

Figure 5: ”hello world!!!” with 8-PAM with Rectangular

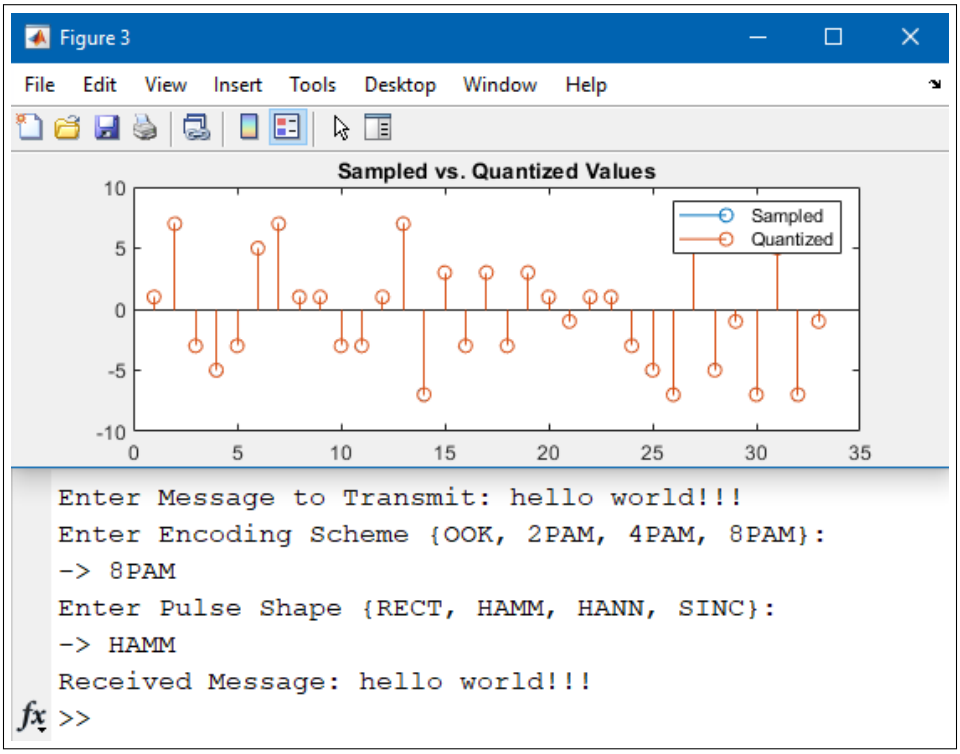

Figure 6: ”hello world!!!” with 8-PAM with Hamming

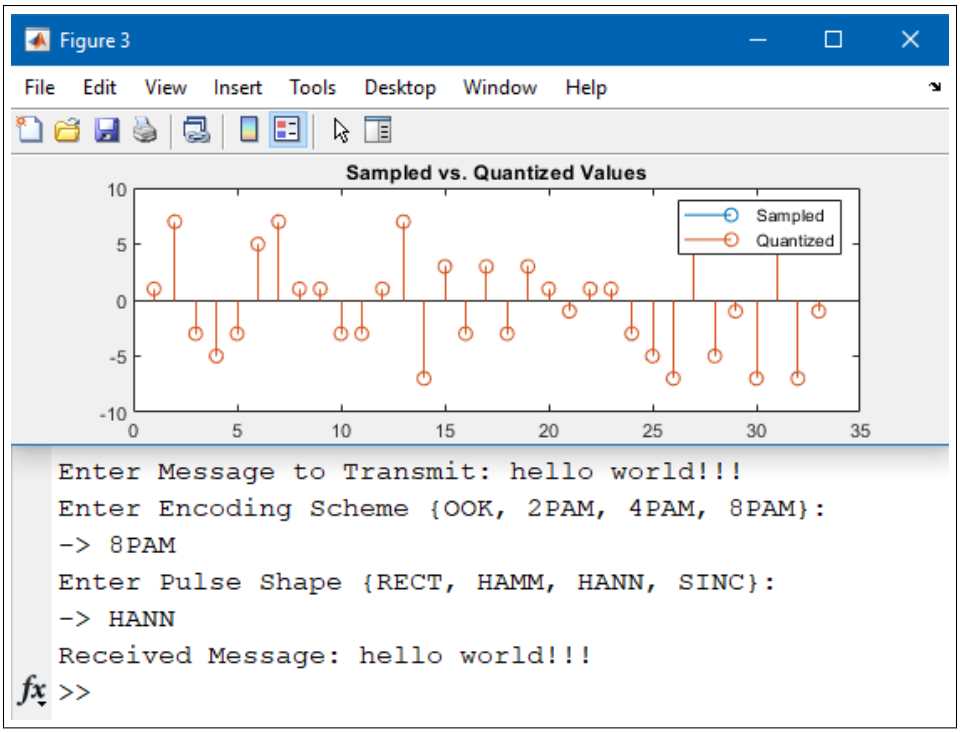

Figure 7: ”hello world!!!” with 8-PAM with Hanning

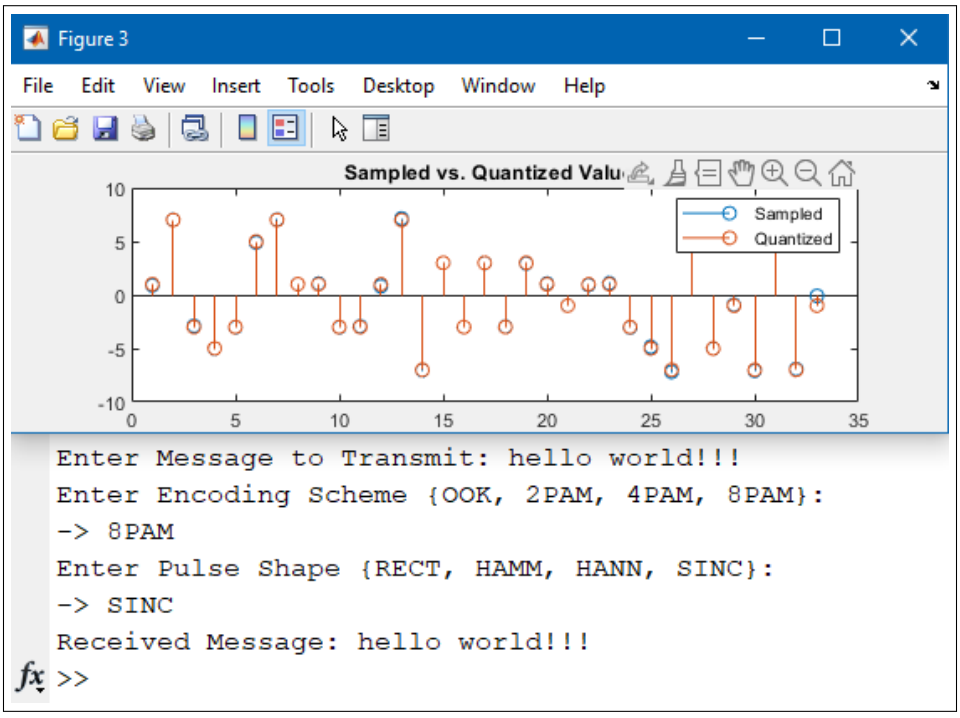

Figure 8: ”hello world!!!” with 8-PAM with Sinc

2.3 Decode the Given Data

Figure 9 display the sampled and the quantized value from the given .mat file with the decoded message shown below the plot.

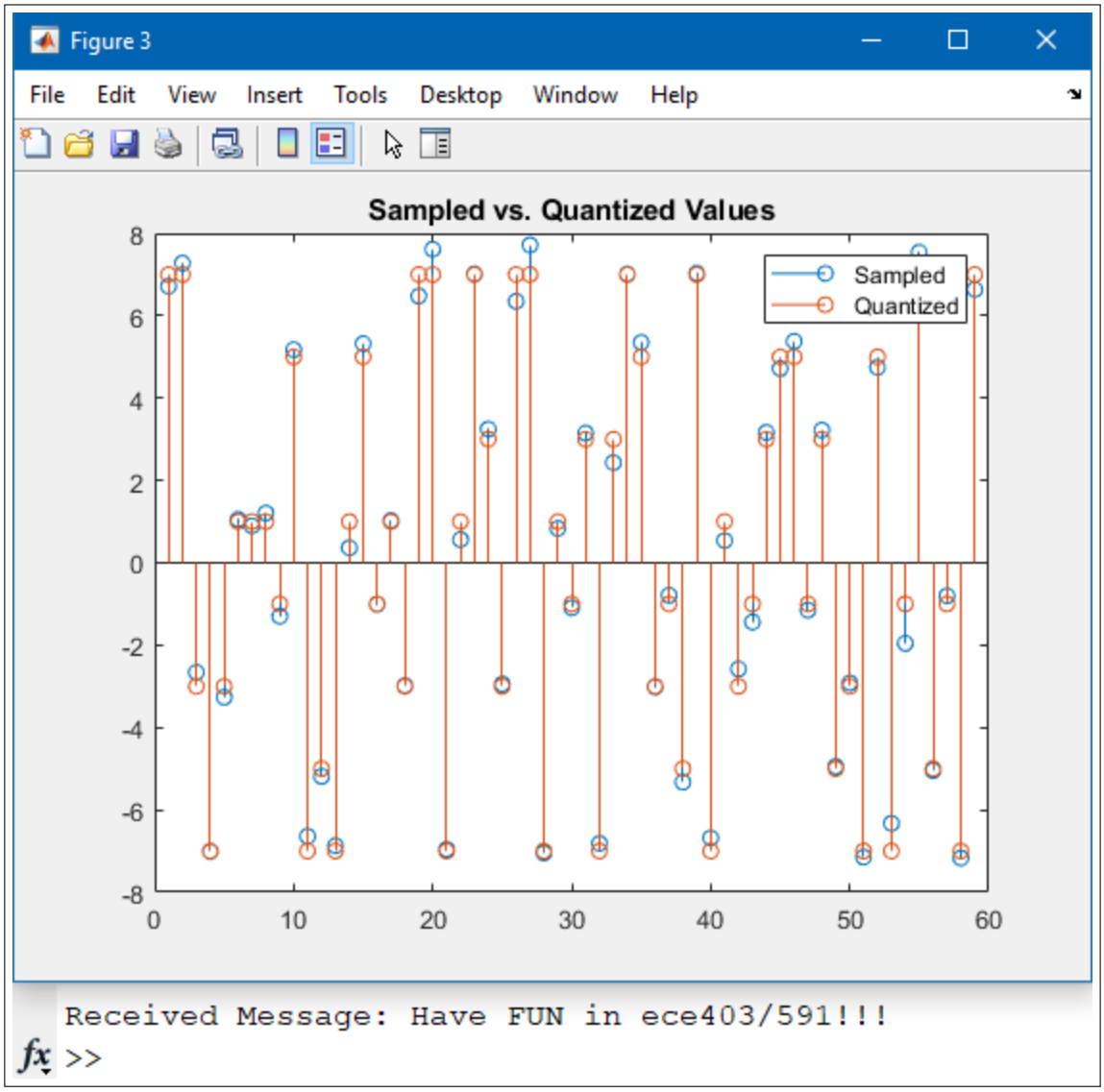

Figure 9: The Given encoded Message

3 Discussion

3.1 Transmitter Design

Designing the ideal transmitter was done in multiple steps. The first step was converting an ascii character string into a binary string, which was handled in $ascii2bin.m$. Then, the selected pulse shape was generated in $PulseShape.m$. Rect pulses were made using the ones command, hamming pulses used the hamming command, hanning pulses used the hann command, all passing sps as the length. Sinc pulses were made using the sinc command, passing a linspace from -6 to 6 of length 12*sps as the argument (to generate the pulse with 5 sidelobes per side). The selected encoding scheme is then passed to $EncodingScheme.m$ where all of the parameters of the different modulation types are specified and the correct function calls for generating the symbols and transmitted signal are returned in the ES object. Then, the input message's corresponding bits are passed through the symbol generation expression, which returns the array of symbols to send. If the number of bits in the message are not an even multiple of bits per symbol, zero bits are added to the end of the message to pad out the last symbol, which will need to be removed on the receive side. This symbol array is then passed through to the signal generation expression, which takes in many arguments and lays each pulse next to each other, multiplying the amplitude of each pulse by the corresponding symbol value. In the case of sinc pulses being used, the output signal is initialized as a zero array, which then has each pulse added to it in the corresponding location. Since each sinc pulse is 12*sps long, and the pulses lay on top of each other, they could not be simply appended on each other like in other pulse shape signal generators. Whichever modulation type was selected determines which $SymbolGen.m$ and $SignalGen.m$ functions get called. Once the standard message signal is generated, the number of samples corresponding to the random time delay is appended to the front of the array with zero values, creating the final transmitted signal with a random delay before the first symbol. Every combination of pulse shape and modulation type transmissions of "hello world" are shown in Figures 1 to 4 and Figures 10 to 21.

Comparing the rectangular pulse shaping baseband transmission to hamming pulse shaping baseband transmission for 8PAM, some major change in the spectrum of the signal can be observed. In the rectangular pulse, it can be observed that there are pulses/peaks that exist throughout the frequency range with the strongest existing in the center frequency where the exponential decay on the strength of the other peaks occur as the frequency move away from the center frequency. For hamming, the signal behavior in very similar way with the difference for no peaks exist around 50Hz away from the center frequency. Hanning behave similar as the Hamming except that there is an additional very small bump (peak) that occur approximately 50Hz away from the center frequency whereas for hamming, it was a smoother wave. When sinc pulse is apply

rather than rectangular, hamming or hanning, the major difference that can be observed easily is the bandwidth of the modulated signal in the spectrum. That for hamming and hanning pulse shaping, the modulated signal bandwidth can be approximately to be around 100Hz, for rectangular it is around 200Hz whereas the bandwidth for sinc is only approximately to be 50Hz. Bandwidths were also measured from the FFT on an OOK modulated message with each pulse shape. The results are shown below in Table 5.

Window Shape	**Bandwidth**
RECT	INF
HAMM	500 Hz
HANN	450 Hz
SINC	150 Hz

Table 5: "hello world!!!" with OOK: Bandwidth for each Pulse Shape

3.2 Receiver Design

Given the frequency, symbol rate, pulse shape and encoding scheme used by the transmitter, as well as the gain and transmission + initial time delays, taking the received signal and decoding the message is relatively trivial. The biggest challenges involved are ensuring samples are taken at the correct times of the signal, and the correct number of bits are discarded at the end of the message to recover only whole ascii characters. Working backwards through the receiver, the first system implemented was converting binary strings to ascii characters. This was implemented in $bin2ascii.m$, which would be fed the output from one of the four decoder functions for each of the modulation types (OOK, PAM2, PAM4, PAM8) implemented in their respective functions based off of $OOK\_Decoder.m$. Within each decoder function the discarding of excess trailing bits is handled, ensuring an even multiple of seven bits is passed to the ascii conversion. Feeding into the decoder functions is the respective quantizer functions, which take samples of the received signal and map them to the closest corresponding symbol value. For example, in 4PAM, a sample of the received signal might be 2.7, which the quantizer would map to the symbol 3, which corresponds to '10' in binary. The boundaries for these decisions were decided to be evenly spaced between each symbol, following the equation below.

$$Q(x) = \begin{cases} 3 & \text{if} \quad 2 < x \\ 1 & \text{if} \quad 0 < x \leq 2 \\ -1 & \text{if} \quad -2 < x \leq 0 \\ -3 & \text{if} \quad x \leq -2 \end{cases} \tag{5}$$

The quantizers receive their inputs from their corresponding sampling methods. For Rect, Hamm, and Hann pulse shapes, the sampling is extremely simple. The index of the first sample is determined by adding the initial time delay and transmission delays, then dividing the result by the sampling period Ts. Then this result is added to $sps/2$, which spits out the index of the center of the first pulse. Each corresponding pulse is sampled at the previous index + sps, which ensures the receiver always samples in the center of each pulse. This process is repeated until the end of the received signal is reached, and the result is the sampled symbols ready to be quantized. For Sinc pulse shapes, the general method for sampling is very similar, but the starting index is the total number of samples for the two time delays plus $6 * sps$ to place the first sample in the center of the first pulse. Since each Sinc pulse has the 5 sidebands per side, with a total length of $12 * sps$, this puts the first sample directly in the middle of the first pulse. Because of the pulse shape being longer, this sampling must also be stopped $6 * sps$ indexes before the end of the signal to avoid sampling bad data.

With the sampling, quantization, decoding, and conversion to ascii characters laid out, the receiver design is complete. Combining this together with the transmitter before, the full system can be tested, resulting in Figures 5 - 8 and Figures 22 - 33. These figures show each iteration of encoding scheme and pulse shape being used to transmit and receive the message “hello world!!!” successfully.

3.3 Decode the Given Data

Having the transmitter and receiver implemented and working as expected, decoding the given data was again pretty trivial, as all the hard to determine parameters were already specified along with the signal in the $trans\_signal.mat$ file. The transmission delay and propagation delay are given in the $delta$ and tau variables, the sampling rate and symbol rate are given in the fs and $symbolrate$ variables, the encoding scheme and pulse shapes are given in $modulation$ and $pulseshape$ variables, and the attenuation is given in $gain$. With this information, the only thing necessary for decoding the data is running the previously designed receiver and specifying each of these options. Doing this results in a decoded message “Have FUN in ece403/591!!!”, as shown in Figure 9

4 Conclusion

Implementing ideal baseband digital transmitters and receivers in matlab was a valuable learning experience to fully understand the relationship between the different modulation types, pulse shapes, and how they affect performance and implementation. From designing each of these pulse shapes and modulation types transmitters and receivers, we can say the hardest part involved in implementing these would be determining what is happening between the transmitter and receiver that for the sake of simplicity in this assignment, we assumed were already known. On the transmitter side, the only "smarts" that could potentially be added would be gain adjustments for strong multipath or too much attenuation through the channel, or increasing/decreasing the rate depending on error rates to ensure messages are transmitted correctly. On the receiver though, in a real implementation, the initial time delay, transmission time delay, and attenuation through the channel would always have to be determined (assuming a mobile receiver with varying position), even if the modulation type, pulse shape, and symbol rate of the transmitted signal were known. These complications make a "real" implementation of a receiver much more difficult that the ideal counterpart we developed here. One method for possibly determining the start of transmission (transmission and initial time delays) could be a simple receive power detection circuit, but this could obviously trigger falsely due to noise or third-party channel traffic. To counteract false triggers, a standard message header would likely be put at the beginning of transmitted messages to indicate valid/invalid packets. To implement attenuation/gain compensation, the receiver could try to function based off of relative amplitudes instead of absolute amplitudes like implemented in this assignment. This might be difficult, and instead an other option would be to implement some form of channel learning after an initial 'lock', so the receiver can try to understand what is happening as things change in the channel and adjust gain to accommodate for changes in attenuation through time. Both of these proposed solutions require much more 'brains' in the receiver than in the ideal design implemented here, and will surely prove to be the difficult problems solved by us as engineers in the future, along with frequency and phase synchronization in the demodulation from carrier frequency.

If the symbol rate, frequency, pulse shape, and encoding scheme were also unknown, and someone intended to decode a given message signal, even without encryption things would become much more complicated. With discrete components for transmitters and receivers, receiving a message not intended for you is not usually very likely, but with software defined radio, the entire spectrum is open to you, and intercepting someone else's messages becomes much more possible. The frequency spectrum can be observed for interesting signals, allowing users to determine center frequencies with relative ease. Then, in the time domain pulse shapes should be observable, as well as a general estimate of encoding schemes and symbol rate. From there, it is a combination of

fine tuning these estimated parameters and determining the other receiver parameters as described before. Compared to developing a discrete receiver specific to the application of intercepting the message, SDR allows anyone to try their hand at seeing things not intended for them, posing a much greater security risk for secure message content and emphasizing the need for encryption on messages sent over the air.

5 References

[1] "ECE 591-02 Software Defined Radio Midterm Project: Baseband Digital Communication Transceiver Design."

6 Appendices

6.1 Transmitter Design

6.1.1 OOK with Various Shaping

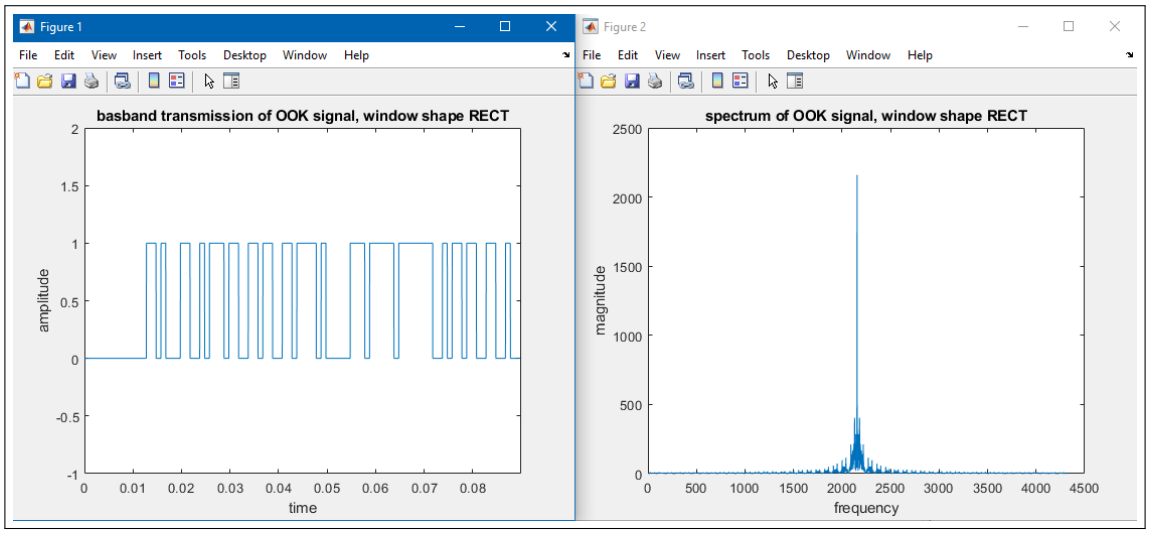

Figure 10: “hello world” with OOK with Rectangular in Time and Frequency Domain

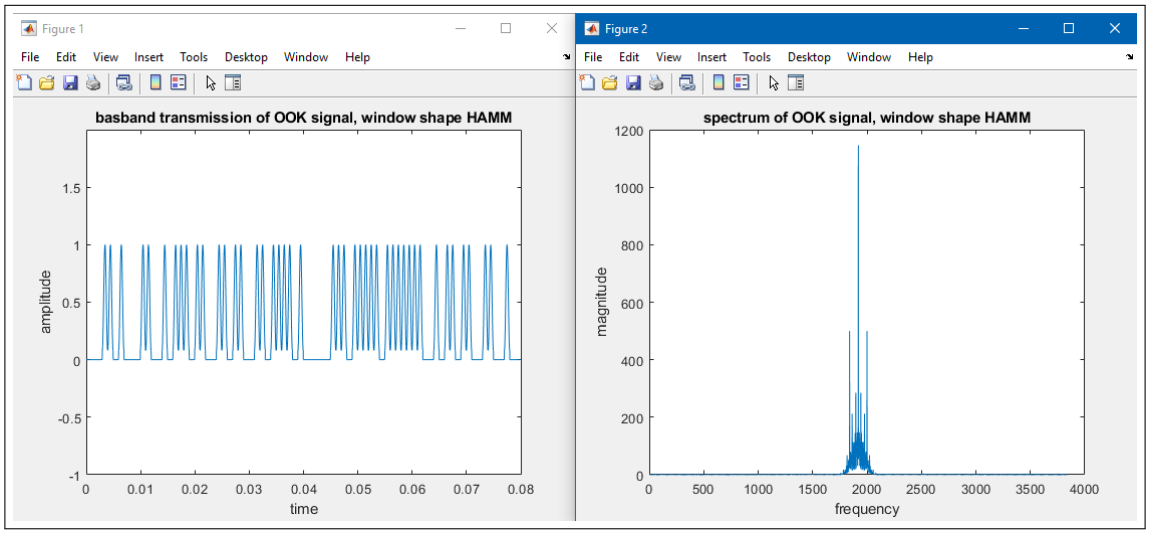

Figure 11: “hello world” with OOK with Hamming in Time and Frequency Domain

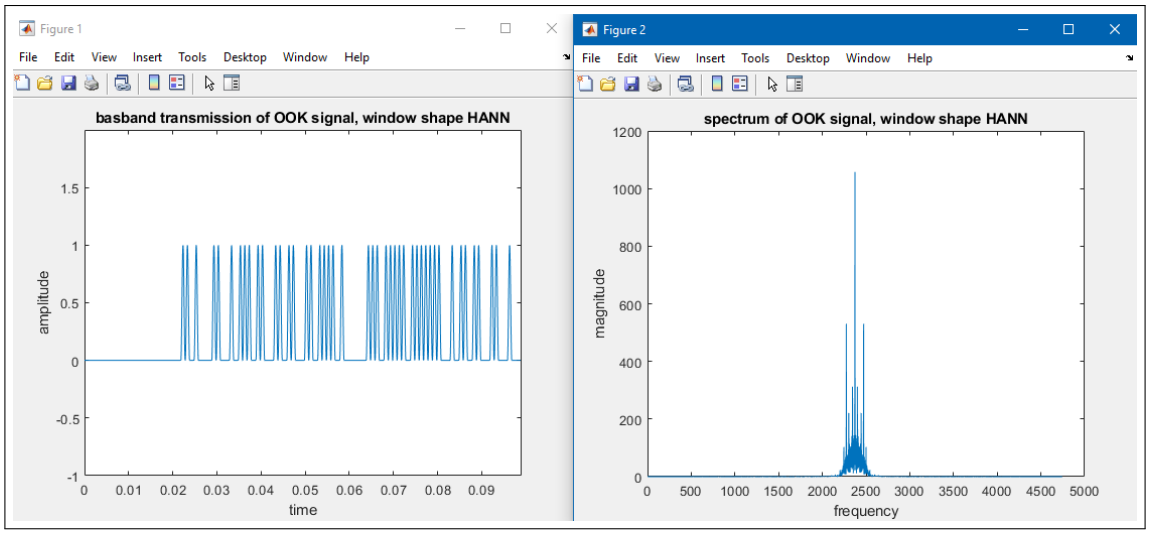

Figure 12: “hello world” with OOK with Hanning in Time and Frequency Domain

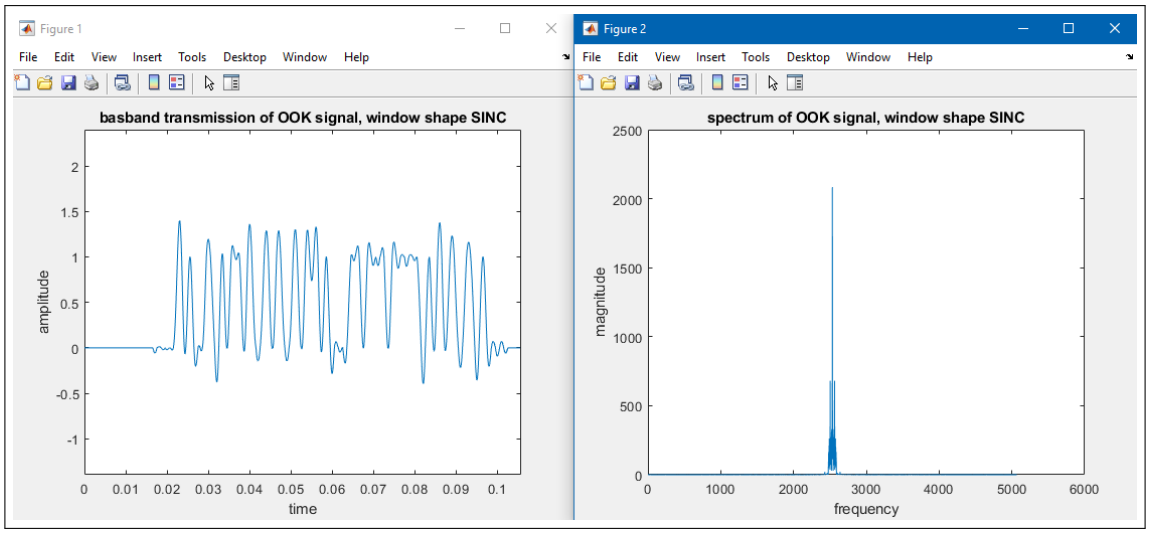

Figure 13: “hello world” with OOK with Sinc in Time and Frequency Domain

6.1.2 2-PAM with Various Shaping

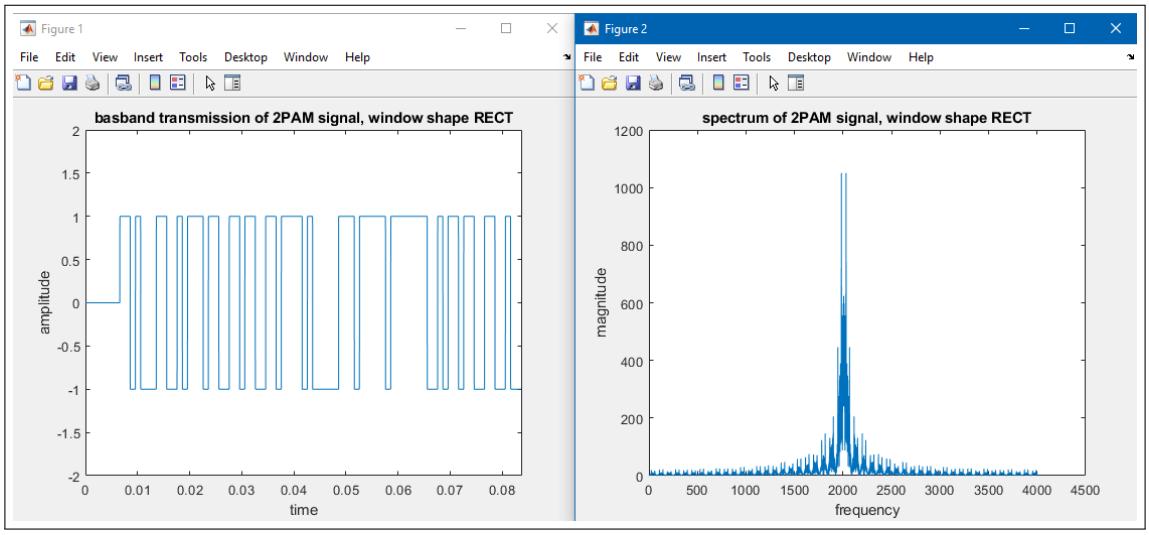

Figure 14: “hello world” with 2-PAM with Rectangular in Time and Frequency Domain

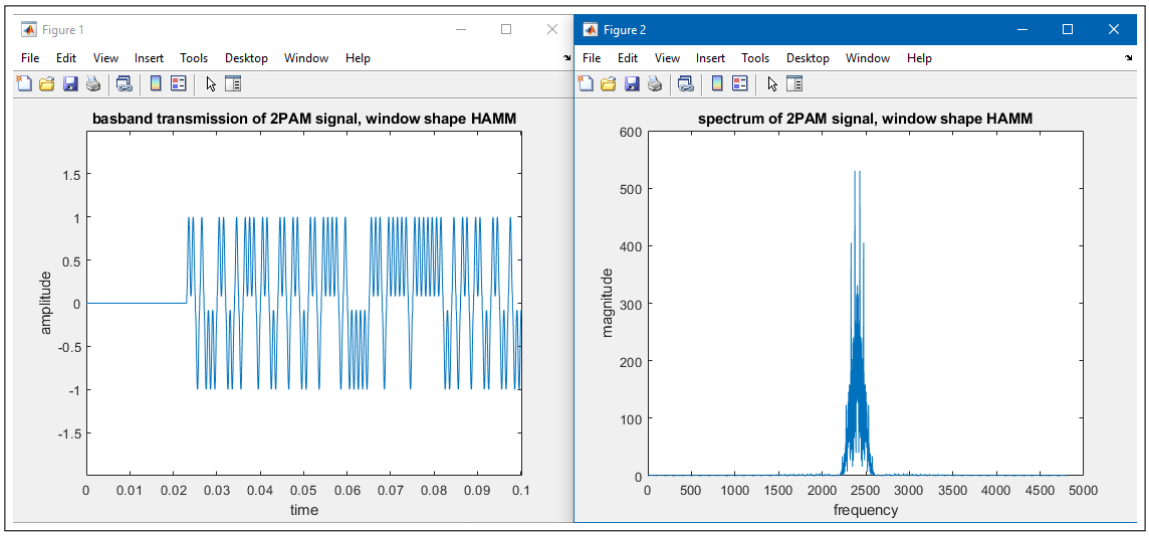

Figure 15: “hello world” with 2-PAM with Hamming in Time and Frequency Domain

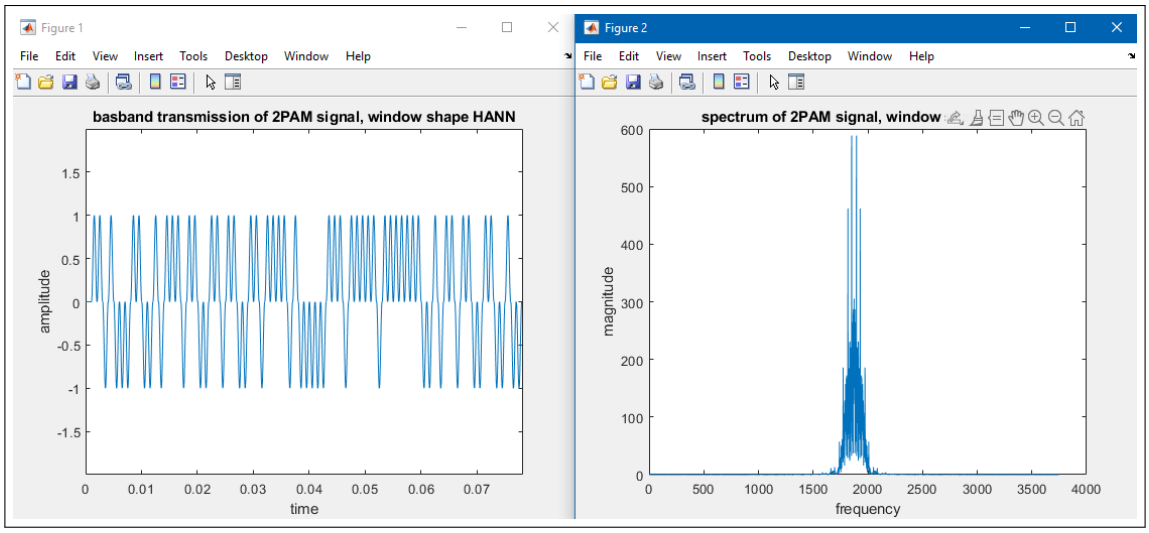

Figure 16: “hello world” with 2-PAM with Hanning in Time and Frequency Domain

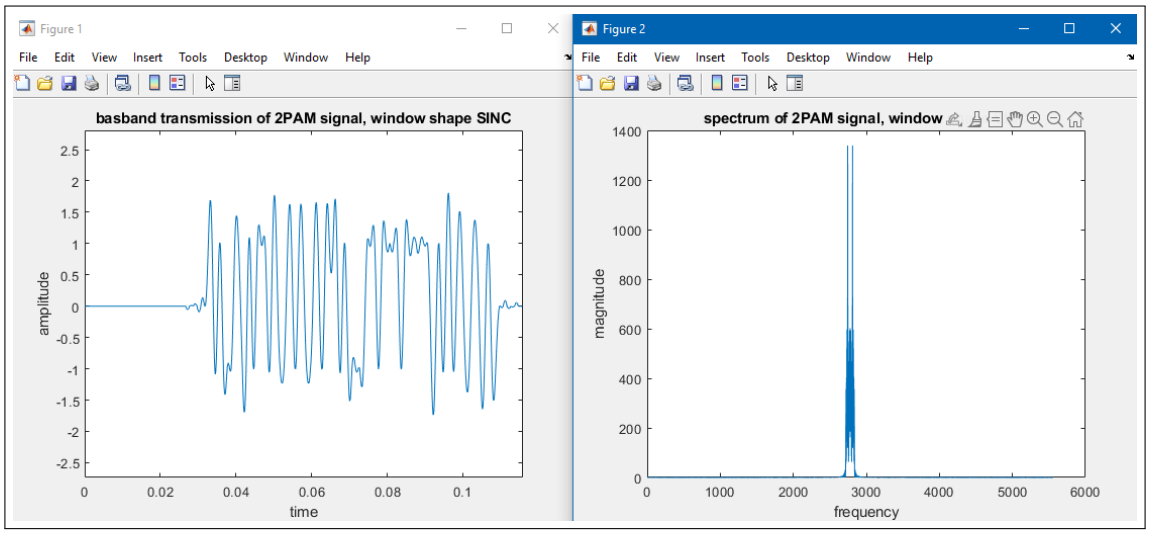

Figure 17: “hello world” with 2-PAM with Sinc in Time and Frequency Domain

6.1.3 4-PAM with Various Shaping

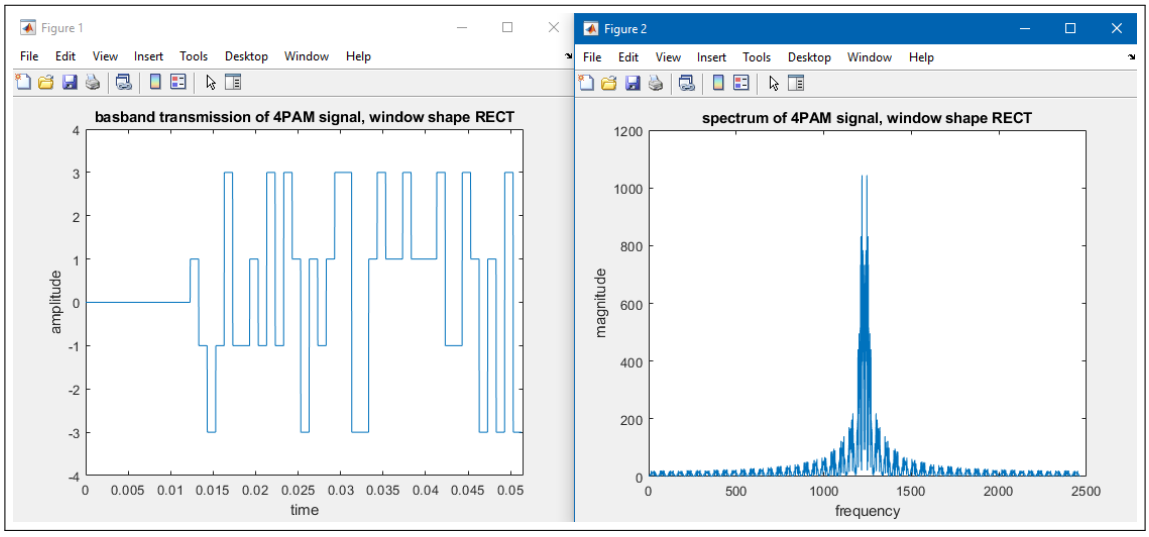

Figure 18: "hello world" with 4-PAM with Rectangular in Time and Frequency Domain

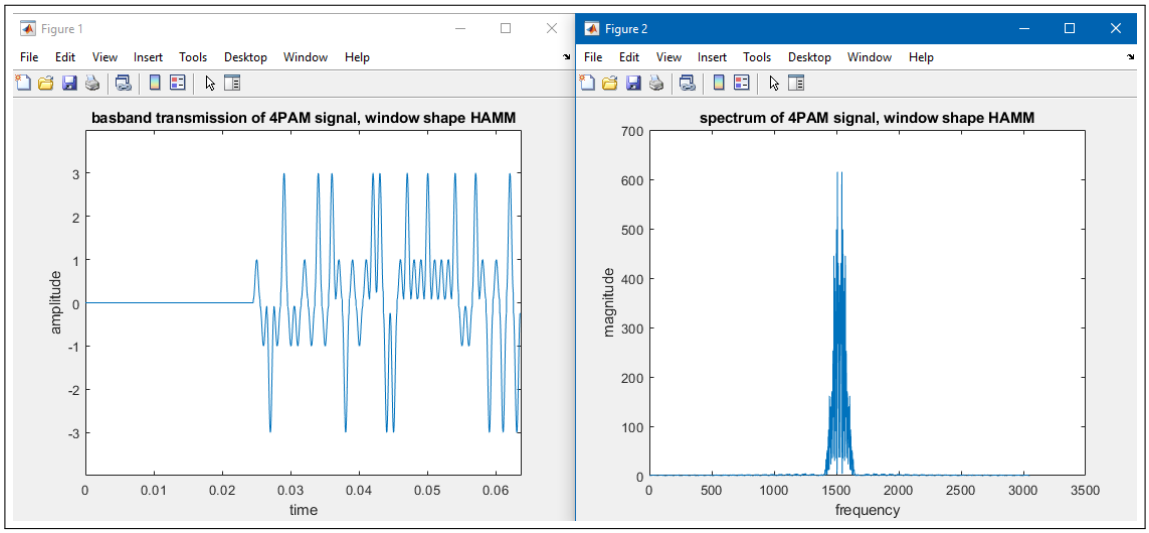

Figure 19: "hello world" with 4-PAM with Hamming in Time and Frequency Domain

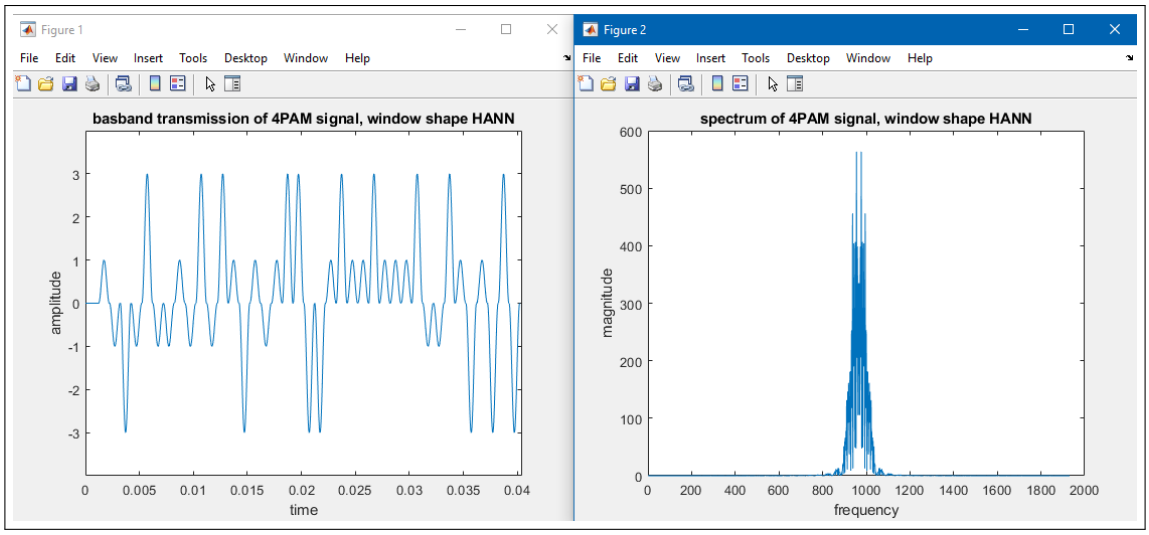

Figure 20: “hello world” with 4-PAM with Hanning in Time and Frequency Domain

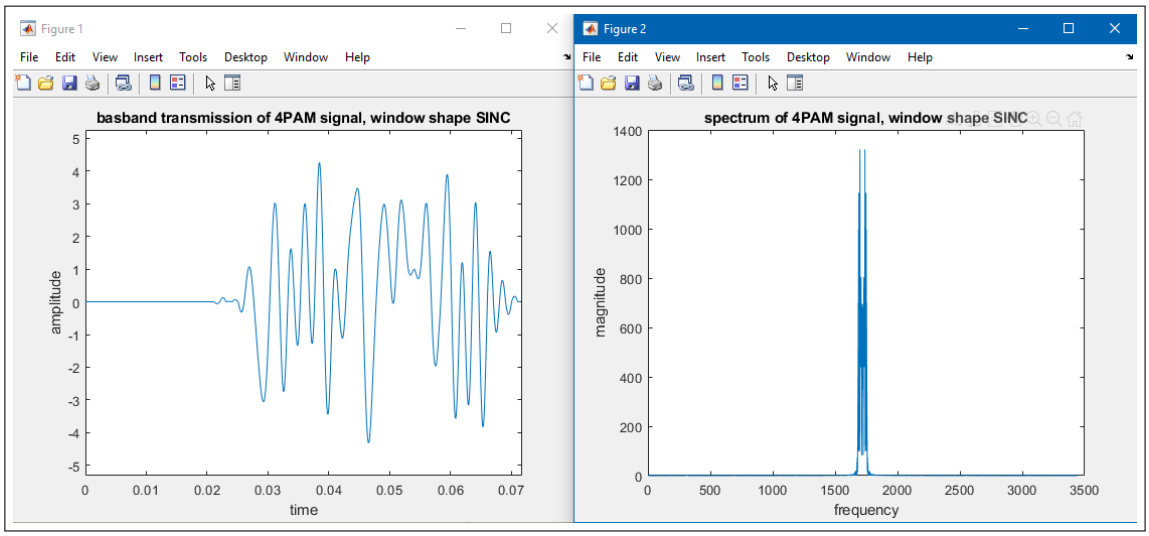

Figure 21: “hello world” with 4-PAM with Sinc in Time and Frequency Domain

6.2 Receiver Design

6.2.1 OOK with Various Shaping

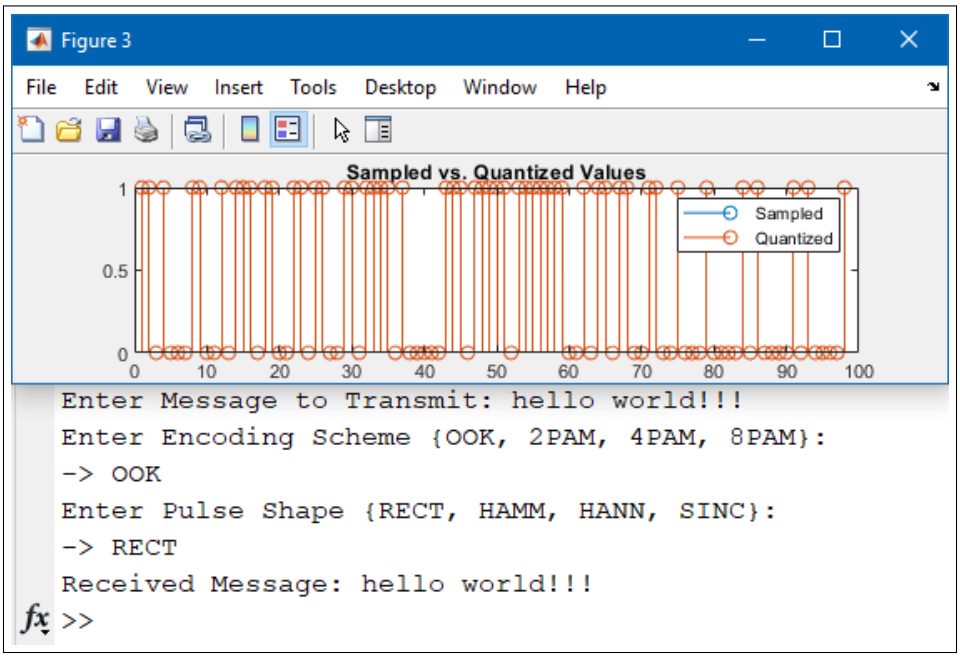

Figure 22: ”hello world!!!” with OOK with Rectangular

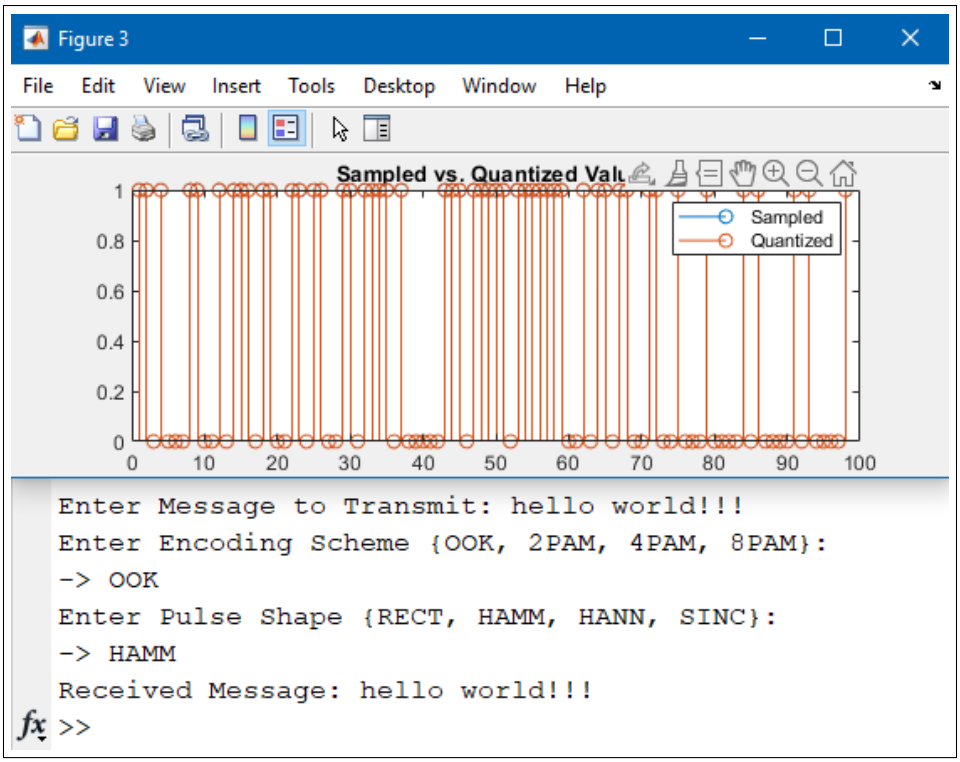

Figure 23: ”hello world!!!” with OOK with Hamming

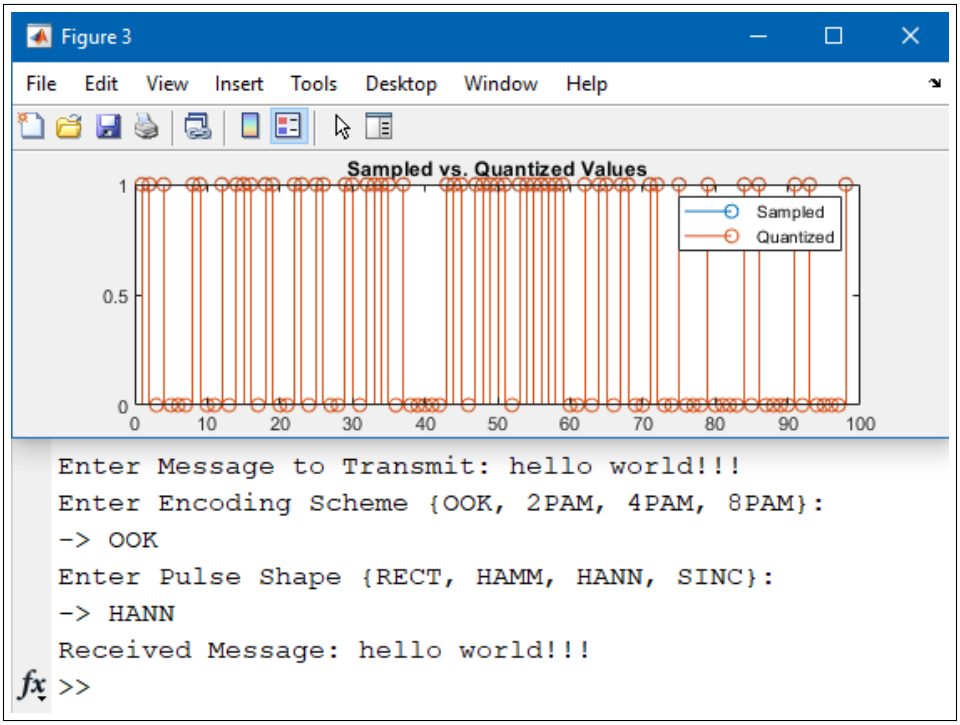

Figure 24: ”hello world!!!” with OOK with Hanning

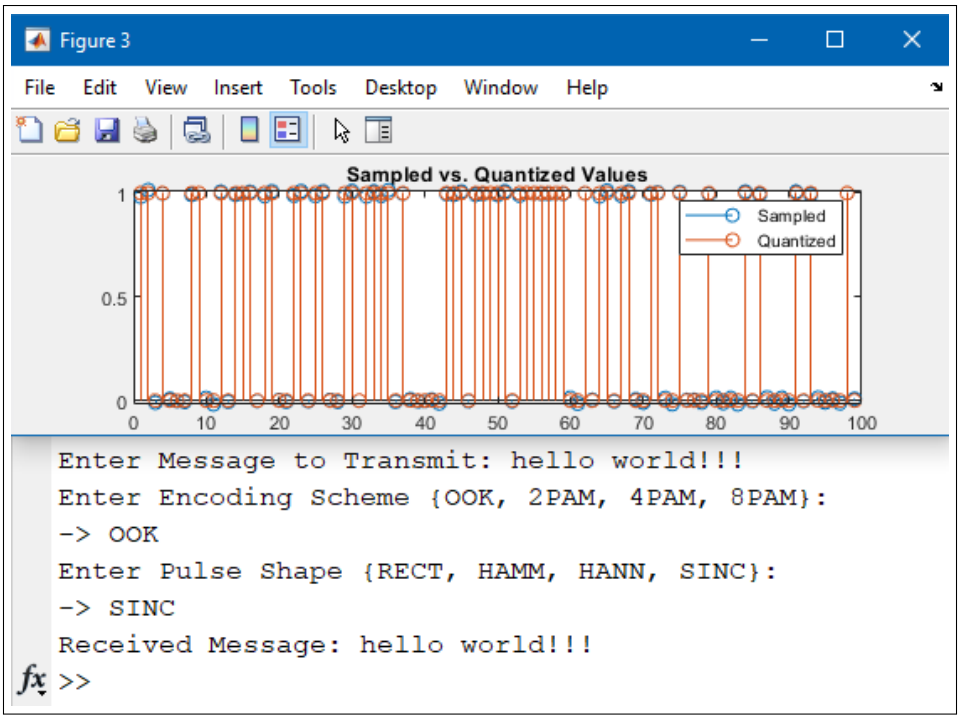

Figure 25: ”hello world!!!” with OOK with Sinc

6.2.2 2-PAM with Various Shaping

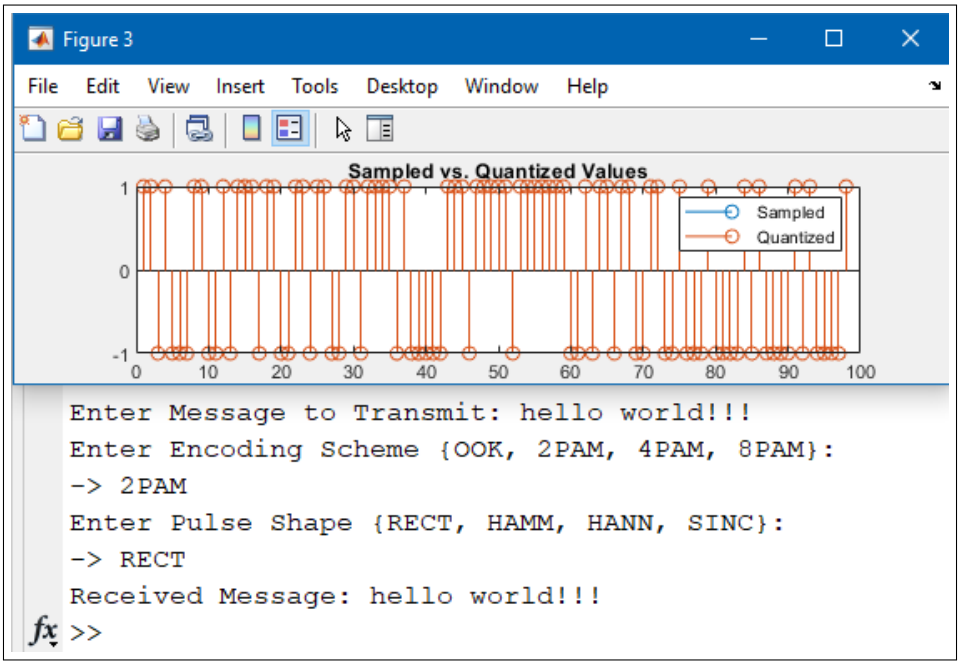

Figure 26: ”hello world!!!” with 2-PAM with Rectangular

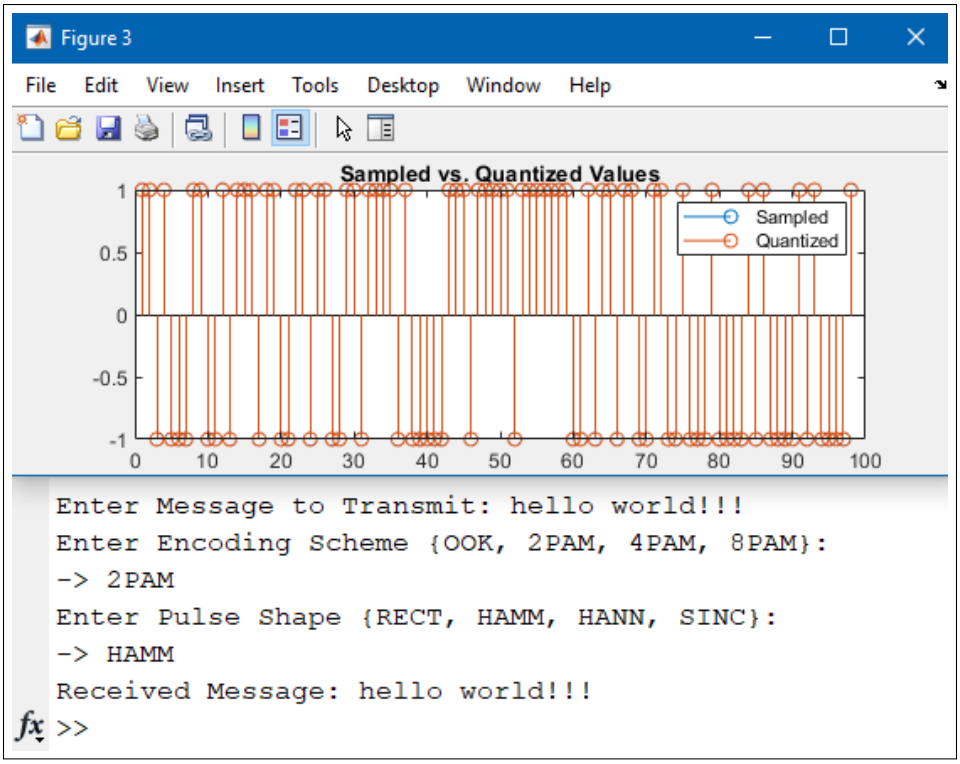

Figure 27: ”hello world!!!” with 2-PAM with Hamming

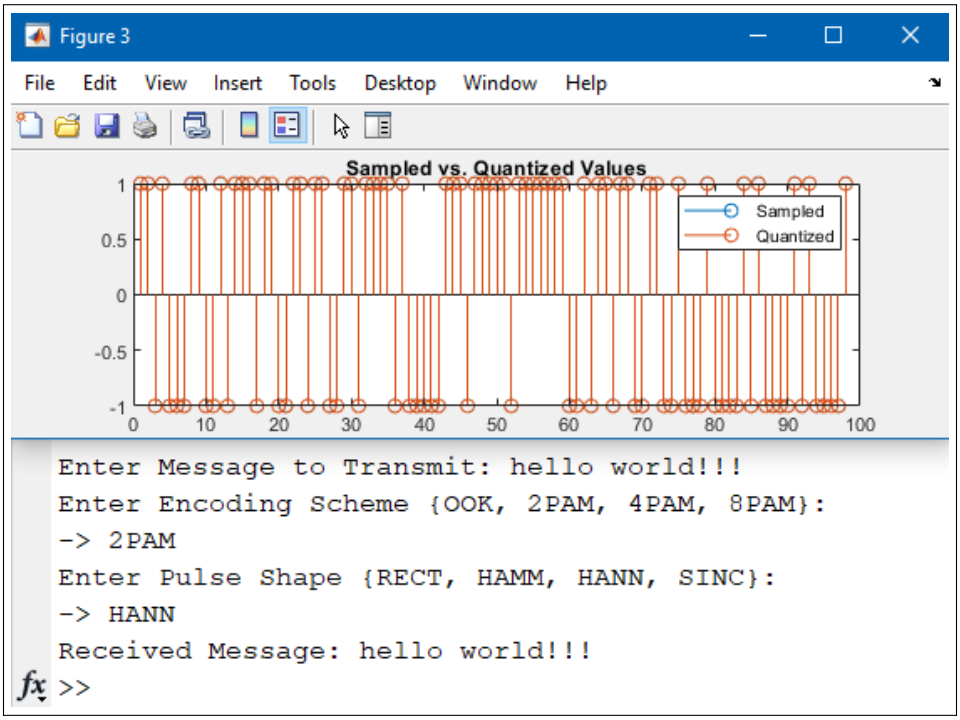

Figure 28: ”hello world!!!” with 2-PAM with Hanning

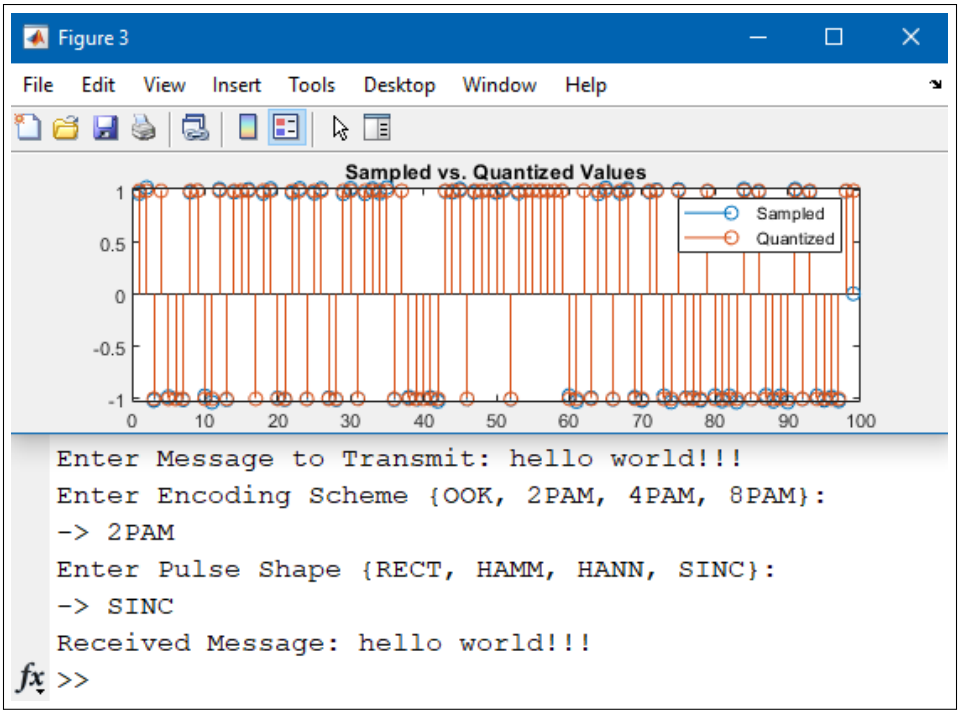

Figure 29: ”hello world!!!” with 2-PAM with Sinc

6.2.3 4-PAM with Various Shaping

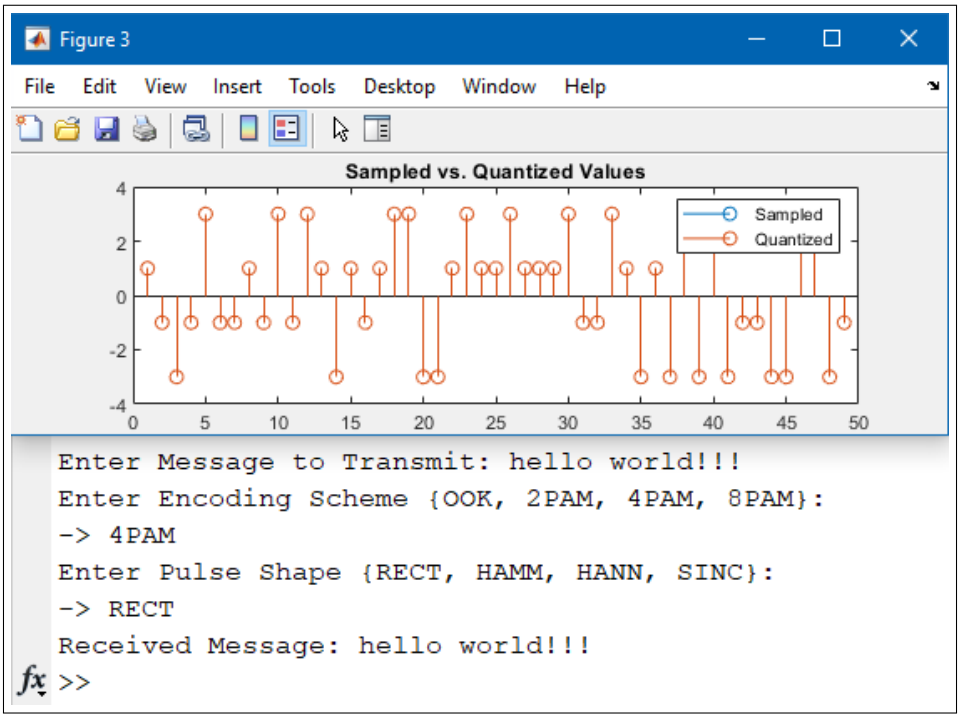

Figure 30: ”hello world!!!” with 4-PAM with Rectangular

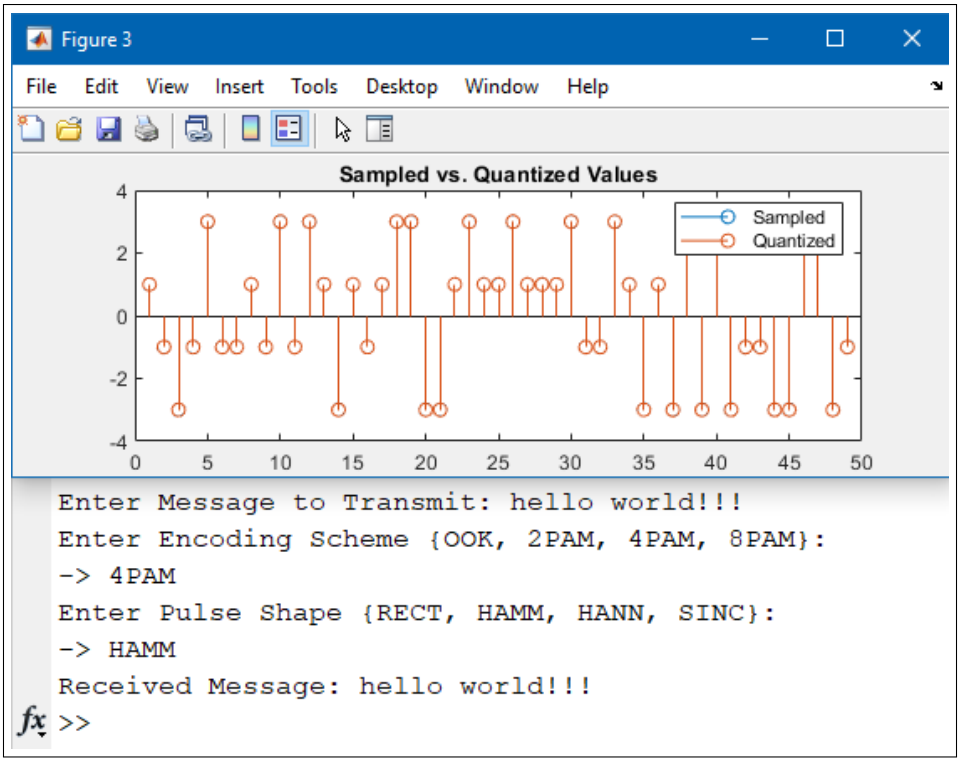

Figure 31: ”hello world!!!” with 4-PAM with Hamming

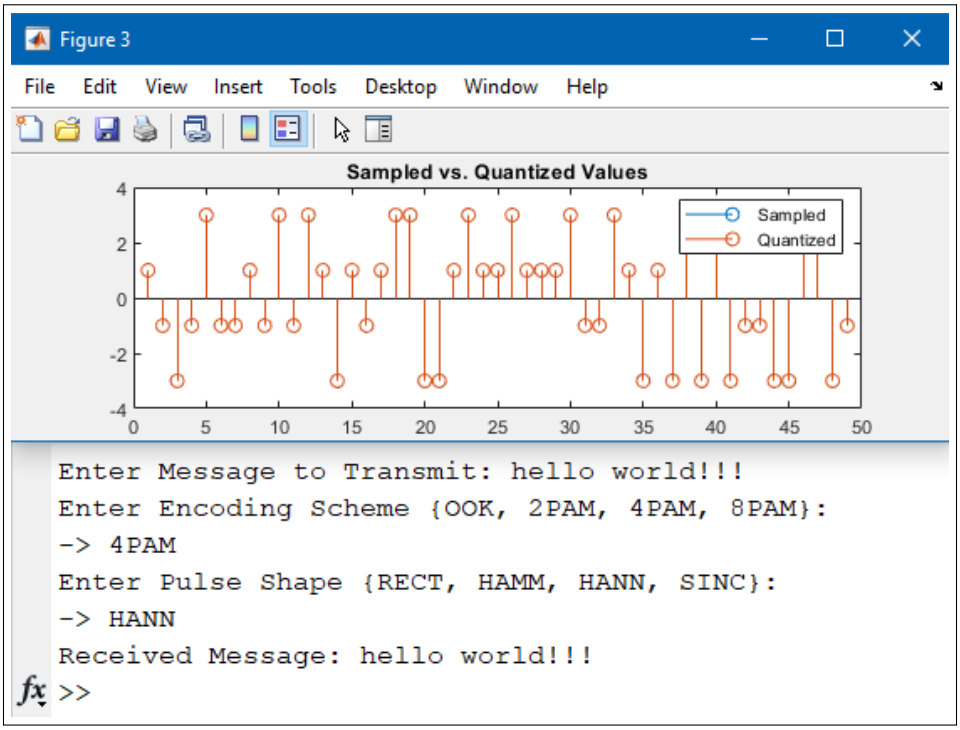

Figure 32: ”hello world!!!” with 4-PAM with Hanning

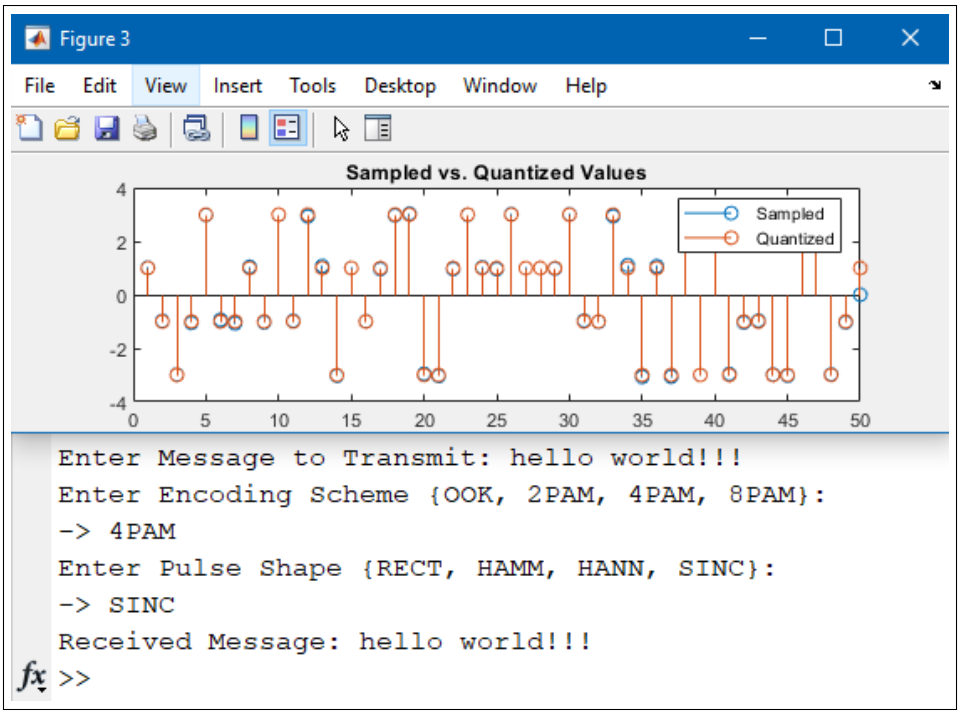

Figure 33: ”hello world!!!” with 4-PAM with Sinc