4-QAM Using Visible Light Communication

Liv Kelley, Junwon Lee, Julian Stone

December 2019

Abstract

In this project, we investigated 4-Quadrature Amplitude Modulation (4-QAM) by demodulating QAM data from a series of small LEDs collected using a photodiode. We used MATLAB to process the data. Specifically, we used cross correlation to find the starting point of the data, then demodulated the data and time synchronized it by raising the data to the 4th power and taking the DTFT. We performed this process at 3 inches and 24 inches distance from the photodiode and encoded the data in a rectangular pulse with a pulse width of 20 and then transitioned to using a root raised cosine at widths 20, 15, 10, and 5. We got a 0 percent error rate for every scenario except the rectangular pulse with pulse width 20 at 24 inches away, which had an error rate of 26 percent. We ran that scenario three times, so we currently theorize that a rectangular pulse with pulse width 20 is more susceptible to noise at 24 inches away. Despite this, we managed to get a bit rate of 400kb/s from the root raised cosine of width 5, so we managed to successfully transmit QAM data at a reasonable data rate.

Introduction

What is QAM?

Quadrature amplitude modulation (QAM) is a form of modulation that allows amplitude modulated message signals to be communicated using phase shifts that are integer multiples of 90 degrees.

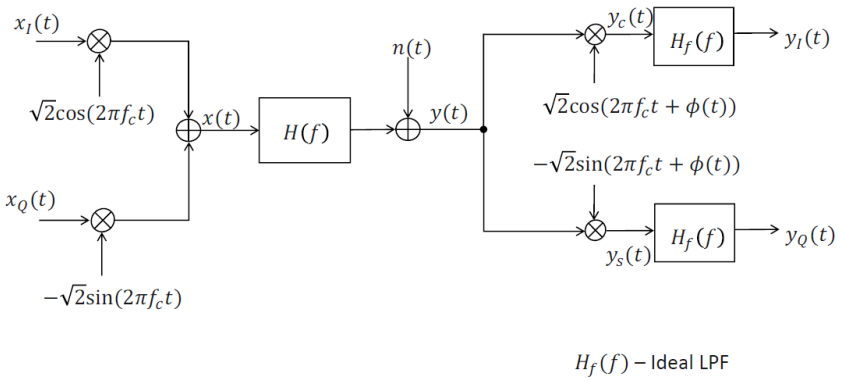


Figure 1: This is a simple representation of QAM. In this diagram, the X signal has already been split up into real and imaginary parts. This represents the transmitter. Then the signal is passed through the system ($H(f)$) and noise is added. Then, on the receiver side, the signal (represented as real and imaginary parts of the output $y(t)$) is passed through another filter. After this it can be sampled (though the sampling process is not represented here).(2)

In 4-QAM, one signal is split up into its real and imaginary component, which will create a sine and cosine signal. The sine and cosine will have a 90 degree offset, so will be in quadrature. The cosine signal can then be named I and the sine signal can be named Q. These signals are added together and then

transmitted. A constellation diagram can be created by plotting the data on an two-axis graph with the real component of each point on the x-axis and the imaginary component of each point on the y-axis. This graph will look like 4 points in a square formation. (1) QAM can therefore be used for a variation of combinations that are multiples of 4. A few of these are 16-QAM, 64-QAM, and 256-QAM. (7)

Transmitting this data will involve passing it through a system ($H(f)$ in Fig.1) and adding some noise (represented by $n(t)$ in Fig.1). The result will be the output $y(t)$ that can be passed into the receiver. The receiver's job will then be to filter the noise from the data using a matched filter ($H_f(f)$ in Fig. 1). In Fig. 1 this is represented by splitting $y(t)$ into the real and imaginary components and then passing it through the matched filter $H_f(f)$. The result is then demodulated and then sampled (not represented in Fig. 1). (5) If successful, the output of the receiver will look the same as the original QAM data transmitted. These configurations can be used to communicate data at different rates and therefore are used in different contexts. (5)

Why is it used?

QAM can support higher data rates than other amplitude modulated schemes. (3) When QAM is used the data throughput, QAM order, transmit power and the forward error correction can be adjusted dynamically to produce the ideal throughput. (7) Unfortunately, there are also downsides to using QAM. A significant downside is that higher order QAM systems make the spots in the constellation closer together making them more susceptible to noise and therefore harder for the receiver to decode the signal properly. (7) Carrier to interference, carrier to noise, and threshold to noise therefore are all factors that are considered when designing QAM systems for high data throughput systems. (10)

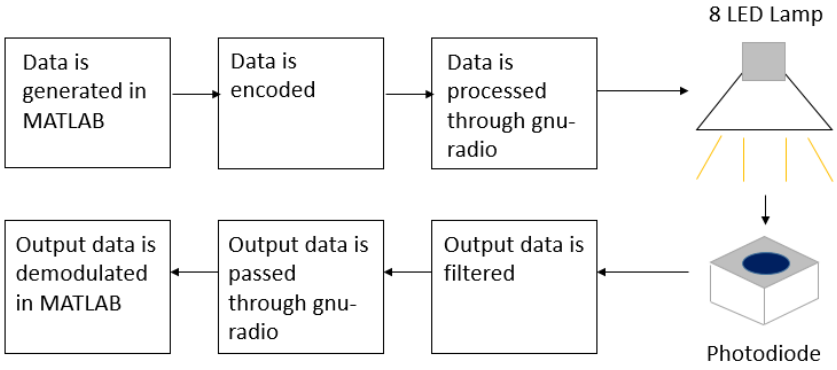
Applications

QAM is used for some radio transmissions, as well as cell phone and wifi communications. (8) (3) Currently, domestic broadcast systems often use 64 and 256-QAM for digital cable radio and for cable modems. (7) Digital Cellular systems in Japan and the United States also used QAM. (6) Finally, an emerging wifi standard called Wi-Fi 6 (802.11ax) was released this year and it is expected to be the dominant standard used by 2021. (9) Wi-Fi 6 includes 1024-QAM as an optional modulation scheme. (9)

Our Implementation

Description of Our System

The figure below shows the diagram of the overall system.



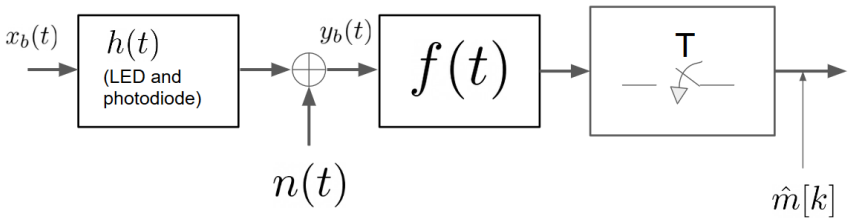
Our set up involves a power generator connected to a lamp with 8 LEDs that can be programmed using gnu-radio to send 4-QAM modulated light pulses that can be received by a photodiode between 5 to 24

inches away. We used gnu-radio to receive and parse the signal from the RX.dat file that the USRP module generated. We then created our own MATLAB program to encode the data either with a rectangular pulse or a root raised cosine, and performed timing synchronization by raising our received data to the 4th power and taking the DTFT. From this result we were able to determine the error rate. The carrier frequency of our system was 250kHz. The sampling rate is 1 MHz. The symbol rate changes, but for the rectangular pulse system it was one symbol per 20 samples, so to get the bit transfer rate we performed the following calculation:

$$1000000MHz \cdot \frac{1}{X} symbol per sample \cdot 2 bits per symbol = \frac{2000000}{X} bits per second$$

where X is the pulse size (20, 15, 10, 5).

Block Diagram



The above block diagram describes the system we are using to 4-QAM modulate and demodulate a signal. The signal goes into the LED and photodiode system ($h(t)$) and then noise is added, which produces $y_b(t)$. $y_b(t)$ then passes through a filter $f(t)$. Essentially, we create $x_b(t)$ in MATLAB by encoding data into pulses, and everything from $h(t)$ to end of $f(t)$ is the physical setup of the system. Once we receive the data, we sample it on MATLAB properly to retrieve data $\hat{m}[k]$, which is a process represented by the switch on the diagram.

Code

The code first generates the data array x where $x[k] = \pm 0.2 \pm 0.2j$. We then encode this to a pulse $p(t)$, where $p(t)$ is either a rectangular pulse or a root raised cosine in our code. Once we encode this, we create a new array tmp, where we store encoded data in a way that the gnu-radio can recognize the data sent. In tmp, all elements in odd position are real values of the data and vice versa. We save this tmp array in 'tx.dat' file, and we use this file to transmit our data through VLC. We used gnu-radio to save the receive data as 'rx.dat' and opened it up on MATLAB. In order to detect the beginning of the data, we found the maximum value of the cross-correlation of the received data and the transmitted data and found its index.

We then collected the section of the RX with maximum cross-correlation and equal length with TX. If our pulse was root raised cosine pulse, we convolved it again with the identical root raised cosine pulse. We see in the constellation that the RX is phase offset, so we had to find f_Δ and ϕ using DTFT. We know that $y[k] = he^{j(f_\Delta k+\theta)}x[k]$ where $x[k] = \pm 0.2 \pm 0.2j$. This is due to the fact that the timing of the transceiver and receiver is off, which creates f_Δ and ϕ. When we normalize it, then $\tilde{y} = e^{j(f_\Delta k+\theta)}x[k]$. When we raise the entire thing to the fourth power, then $(\tilde{y})^4 = (e^{j(f_\Delta k+\theta)}x[k])^4 = -e^{4j(f_\Delta k+\theta)}$. When we run the DTFT of this, we get $DTFT(\tilde{y})^4 = e^{j4\phi}\delta(\Omega - 4f_\Delta)$. By finding the location of impulse and the phase of the amplitude, we find f_Δ and ϕ. Using these values, we divide $y[k]$ with $e^{j(f_\Delta k+\theta)}$. This way, we get $x[k]$. Once we do this, we rotate the constellation by 45 degrees until we reach the full rotation or we find the spot with minimum error rate.

Changes for Efficiency

In the first half of our project, we convolved the original message signal with a rectangular pulse of 20 samples. This was done because when we receive the signal and factor out all of the changes that the signal went through, we can downsample by simply plucking a value in the middle of each pulse. 20 samples is a fair amount of wiggle room to be able to grab a value within the pulse to estimate the original sample at that time. But, the problem with this is that doing this reduces our data transfer rate. Until this point, we had the transmitter sending at a rate of 1 MHz (1 million samples per second). We also were allocating 20 samples per every symbol (each complex number in 4-QAM) that we wanted to send. Also, each symbol actually represents two different bits. So our total transfer rate in bits per second is $1000000(\frac{1}{20})(2) = 100000$. Now we have a way of reducing this if we are smarter about what we convolve the original message with.

A root raised cosine would allow us to reduce inter-symbol interference and simply have a peak that can be detected rather than a large set of relatively equal points to choose from, vastly reducing the number of samples needed. Now to implement a raised cosine, we can convolve the input message with a root-raised cosine impulse response and then convolve the received signal with a root-raised cosine. This has the effect of acting as a matched filter, creating an overall change matching that of a normal raised cosine. Now, after taking care of the effects of the transfer process, we should have distinct peaks that represent the original message at each time step. Since we have distinct peaks, we can vastly reduce the size of the raised cosine signal to increase our data transfer rate. The way we downsample the data now is by first downsampling the data with different time offsets. These downsamples will be offset differently from one another so that no two downsamples have the same points and there should be the same number of downsamples as the pulse size. Now, for each downsampled data, we can calculate the average error. Whichever downsampling process that has the smallest error is the correct downsampling method as it is the one containing the peaks.

Transmitted and Received Signals

Below was the standard constellation for all of the tests we conducted. Next to it is a small sample of the real component of the data. As you can see, the constellation has distinct points at the corners of a square with sides .2 and is centered on the origin. The real component graph displays the random nature of the our transmitted data.

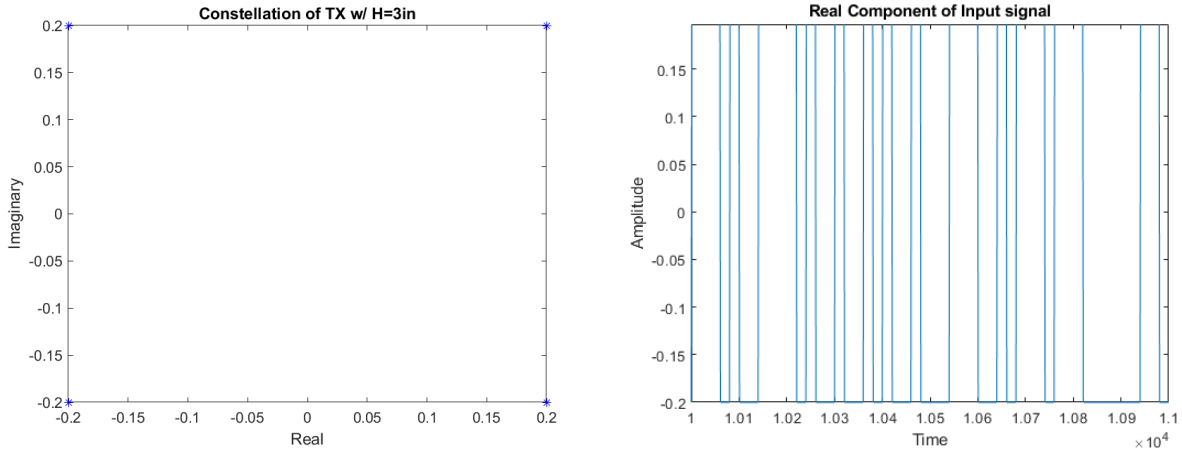


Figure 2: Representations of the data to be transmitted as either a constellation (left) or the real component vs time (right).

Upon receiving the signals we get graphs that look like the samples below. When convolving with a rectangular pulse, we get a real component that looks as it does on the left. When convolving with a root-

raised cosine on both the transmitted and received signals, we get more distinct peaks as shown in the graph on the right.

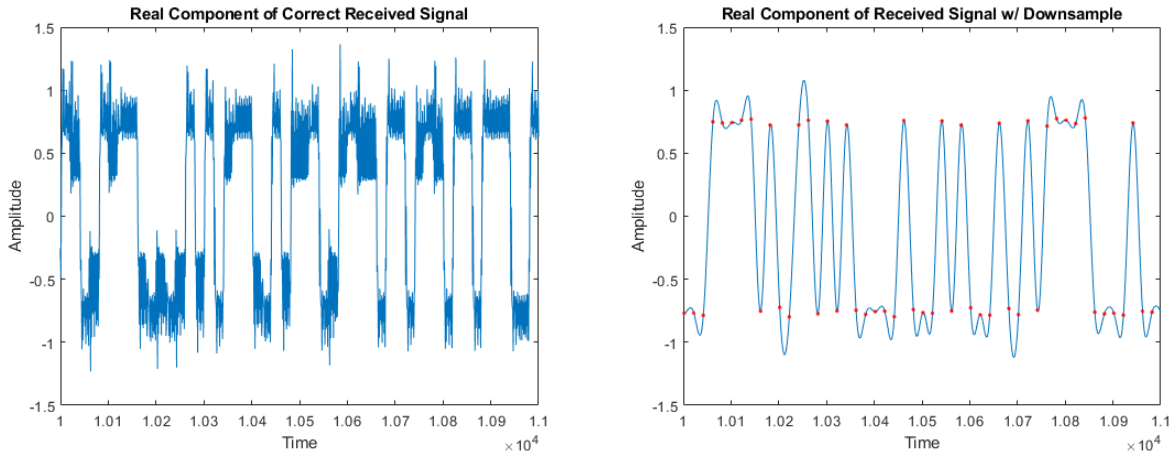


Figure 3: Examples of received signals after correction for both rectangular pulse (left) and root-raised cosine (right) pulses.

Results

Data/Graphs

To begin, we should look at the constellations of the received data before it was corrected and downsampled. These are simply examples, but look very similar for each test. The constellation on the left shows the pre-correction constellation when convolving with a rectangular pulse whereas the constellation on the right shown the pre-correction constellation when convolving with a root-raised cosine pulse.

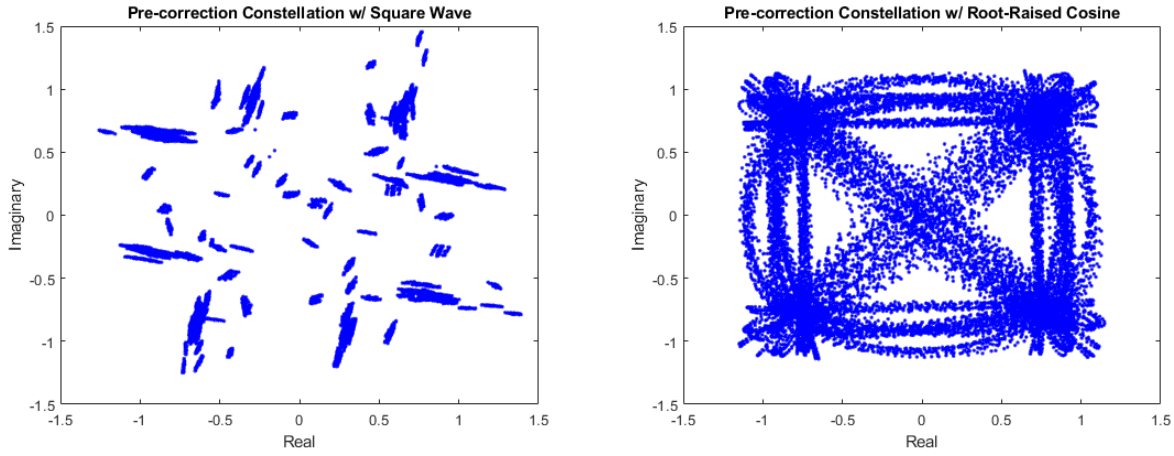


Figure 4: Pre-correction (including time synchronization) and downsampling constellation examples for both rectangular pulse (left) and root-raised cosine (right) pulses.

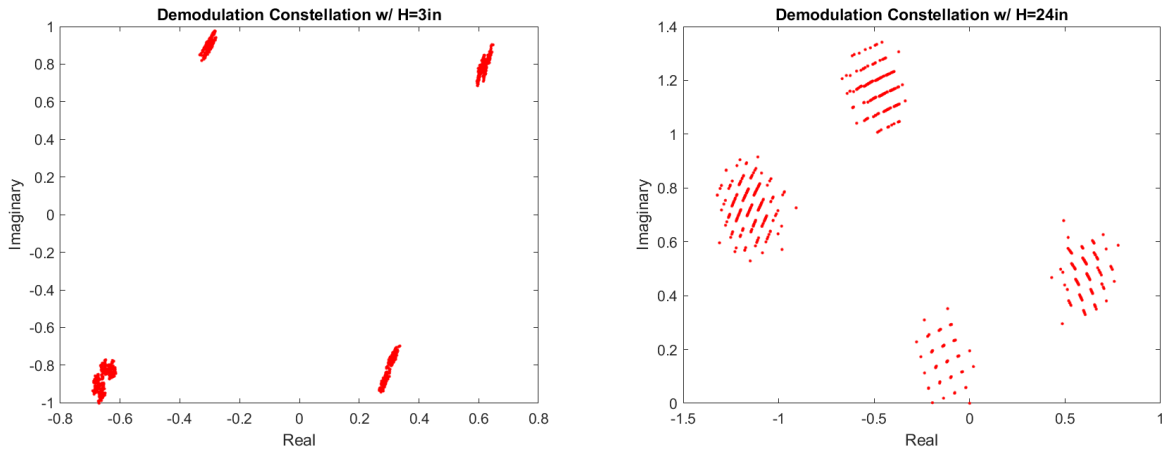


Figure 5: Constellation of received data where the square pulse had the width of 20 at distances 3 inches and 24 inches from the photodiode.

In the demodulated constellation with square pulse where the photodiode is positioned 3 inches from the light, the clusters are clear in four distinct quadrants. However, the points are not clearly centered around the four corners of a square. This is even more evident when we pull up the lamp up to 24 inches, where the clusters are even more off. One possible explanation for this is that when the lamp is forced to shine at higher voltage for a longer period of time, it saturates and escapes away from the required voltage and slowly decays down, which may explain why they are off from the original corners. With the successful data retrieval, we get data rate of 100k bits per second.

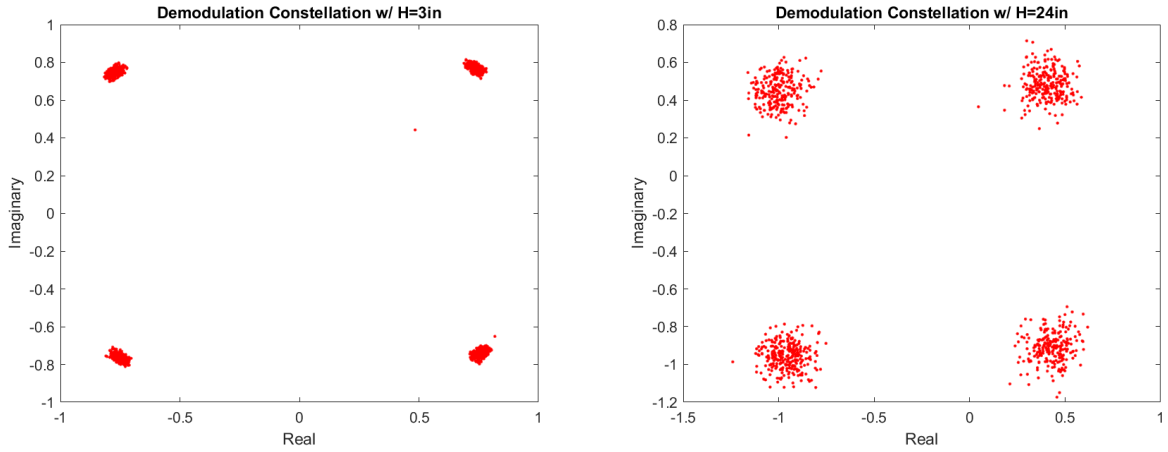


Figure 6: Constellation of received data where the Root-Raised Cosine pulse had the width of 20 at distances 3 inches and 24 inches from the photodiode.

Now that we had a decent test with a rectangular pulse, we decided to move to implementing a root-raised cosine convolution on both the transmitted and received data. We kept the pulse at a width of 20 before moving to smaller widths. As shown above, for both distances, the data is a lot more precise and accurate than compared to that of the rectangular pulse. For both distances the error rate was 0%. When it comes to the bit transfer rate, the value stayed at 100 kb/s as we had not modified the width from the previous test.

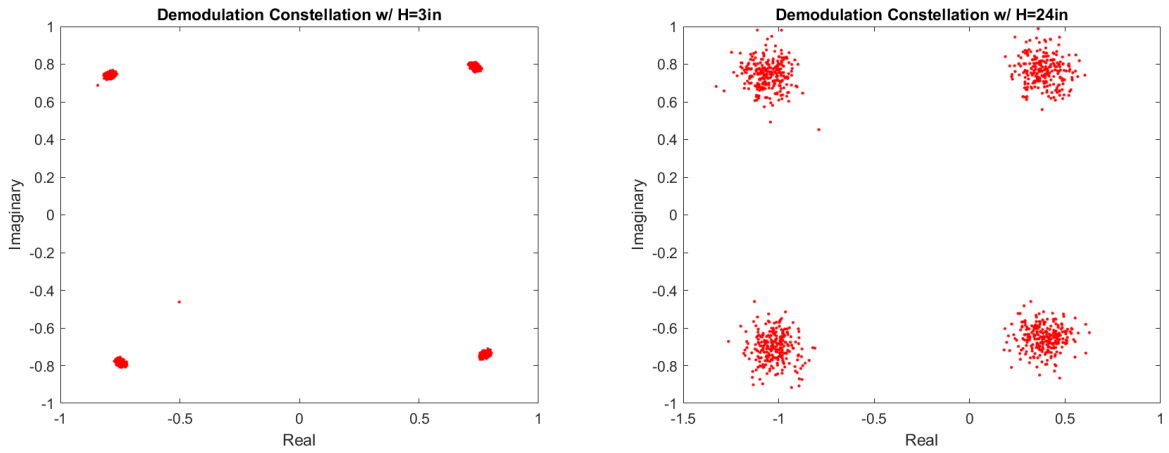


Figure 7: Constellation of received data where the Root-Raised Cosine pulse had the width of 15 at distances 3 inches and 24 inches from the photodiode.

Now that we had the root-raised cosine worked, we moved to reducing the size of the pulse. Now we have a width of 15, a 25% decrease. As shown in the above graphs, very little changes in terms of the constellations. The 24-in constellation is possible a little more dispersed, but still in obvious clusters. This is further validated by the fact that for both distances, once again, the error rate was 0%. Also, with a pulse size of 15, we have a bit transfer rate of ≈ 133.33 kb/s.

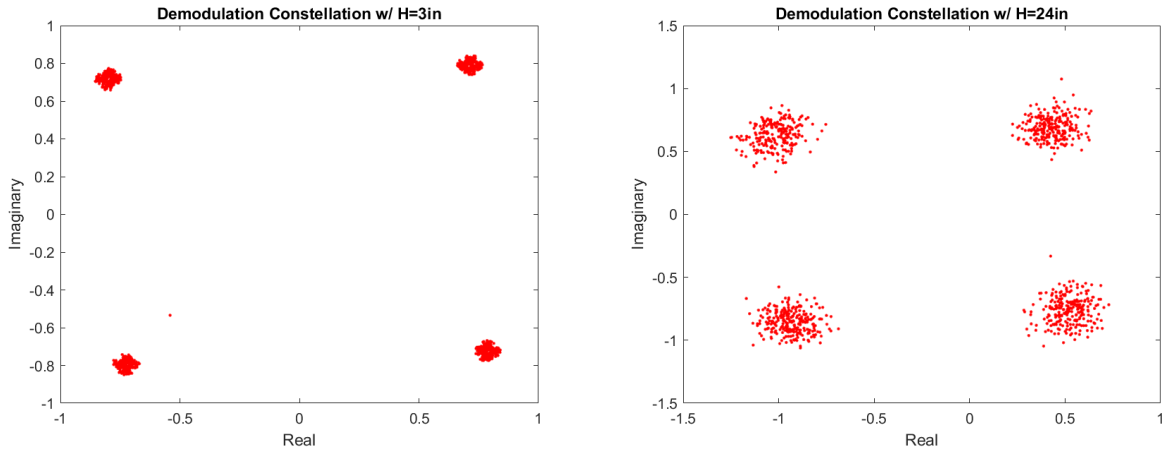


Figure 8: Constellation of received data where the Root-Raised Cosine pulse had the width of 10 at distances 3 inches and 24 inches from the photodiode.

From here we pushed further on the width by reducing it to 10, a decrease of 50% from the original width. Similar to the 15-width data, we can see here that the points are slightly more dispersed, but in no way unable to be interpreted. The data very much remains in four distinct clusters. Once again, this is validated by the fact that there is an error rate of 0%. Now, with a pulse size of 10, we have a bit transfer rate of 200 kb/s, double the initial rate.

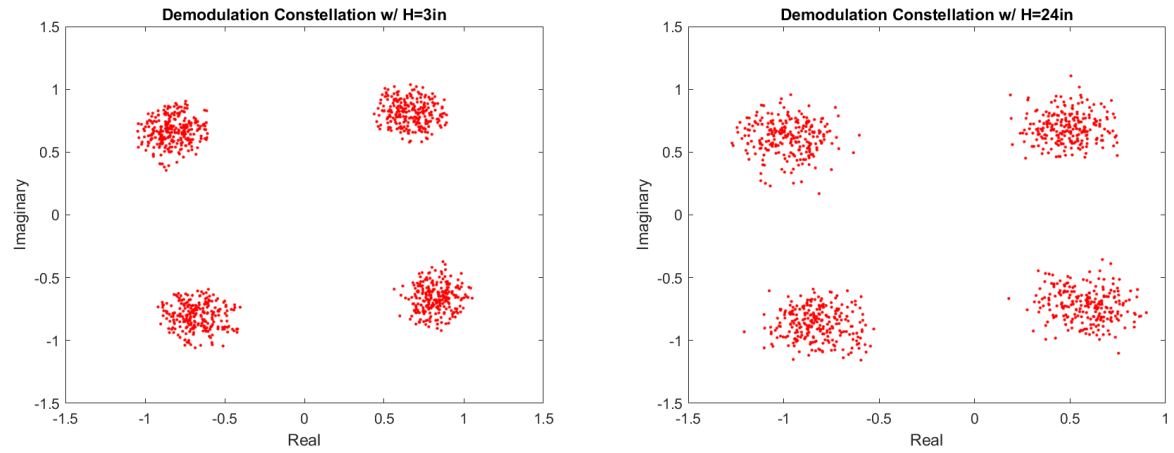


Figure 9: Constellation of received data where the Root-Raised Cosine pulse had the width of 5 at distances 3 inches and 24 inches from the photodiode.

The last push was reducing the pulse size down to 5. We found that the clusters were still clear in four quadrants when the lamp was both 3 inches and 24 inches away from the photodiode. The error rate was still 0 despite the reduced size of the pulse width, which means that we successfully achieved a higher bit transfer rate of 400 kb/s, which is 4 times the starting bit transfer rate.

Conclusion

Summary

As we have shown, we were able to successfully demodulate 4-QAM data transmitted via light in most scenarios. We also found that when sending messages through visible light with the setup provided, it seems best to use a root-raised cosine pulse over a rectangular pulse. It provides more consistent zero ISI and error rate and can be operated at higher frequencies will still maintaining nearly all of the data. Also, the setup can only handle a rectangular pulse up to a certain distance from the photodiode, so a root-raised cosine provides more integration with the system.

Reflection and Suggestions for Improvement

We were pretty happy with the results of our demodulation process. We were clearly able to see the -1s and 1s we transmitted in the results we obtained from the demodulation with an error rate of 0.

We were able to do this at a transmission rate of 400 kb/s, which is very impressive given our initial bit transfer rate of 100 kb/s. If we were to expand on this process we would try to eliminate the remaining error as much as possible, try to decode more complicated messages, and test the quality of signal when there is a larger distance between the LEDs and photodiode.

Who Did What

Everyone worked on everything.

Appendix

GitHub link: https://github.com/junwonlee5/ADC_VLC_Final

Code:

```
clear all
load data_rr_final.mat
y = y(100:end);
[r lags] = xcorr(y, x_pul);

%%
% Collect the section of RX with MAX cross correlation and equal length with TX
[val, idx] = max(abs(r));
% Use next line for root-raised cosine
y_mod = y(lags(idx)+10*pulse_size:lags(idx)+10*pulse_size+1000*pulse_size);
% Use next line for square wave
%y_mod = y(lags(idx):lags(idx)+1000*pulse_size);
% Only use next line for square wave
y_mod = conv(y_mod, pulse);

%%
% Perform DTFT to find the f_delta and phase offset
y_norm = y_mod ./rms(y_mod);
y_2= y_norm.^4;
[X, f] = plot_DTFT(y_2);
[val2, idx2] = max(abs(X));
f_loc = f(idx2)/4; %frequency offset
phase = (angle(X(idx2))+pi)/4; %phase offset

%%
%Correct phase offset from RX
arr = 0:length(y_norm)-1;
exp_corr = exp(1j*(f_loc*floor((arr-1)/pulse_size)+phase));
y_norm_t = y_norm.';
x_rec = y_norm_t ./exp_corr;

ind = 0;
mag = 0;
for a = 1:pulse_size
    m = mean(abs(x_rec(a:pulse_size:end)));
    if(m > mag)
        ind = a;
        mag = m;
    end
end
x_rec_corr = x_rec(ind:pulse_size:end);
x_rec_corr = x_rec_corr(11:end-10);
% Only use next line for square wave
%x_rec_corr = x_rec(10:20:end)
%%
f1 = figure
plot(real(data), imag(data), 'b*')
xlabel('Real')
ylabel('Imaginary')
title('Constellation of TX w/ H=3in')
f2 = figure
plot(real(x_rec_corr),imag(x_rec_corr),'r.')
```

```
xlabel('Real')
ylabel('Imaginary')
title('Demodulation Constellation w/ H=3in')
%%
x_com = [];
p_err = 1;
ph = 0;
for a = 0:15
    x_rec_corr_rot = x_rec_corr*exp(1j*a*pi/8);
    x_compare = [];
    x_compare = sign(real(x_rec_corr_rot))+1j*sign(imag(x_rec_corr_rot));
    x_compare = .2*x_compare.';
    err = x_compare - data;
    num_err = length(find(err~=0));
    error = num_err/length(data);
    if(error < p_err)
        p_err = error;
        x_com = x_compare;
        ph = a;
    end
end
```

References

[1] *Govindasamy, S.* (2019, December). Analog and Digitial Communications Lecture 19. Analog and Digtial Communications Class. Olin College of Engineering. Needham, MA.

[2] *Govindasamy, S.* (2019, December). Carrier Synchronization: Digital Costas' Loop Approach. Analog and Digtial Communications Class. Olin College of Engineering. Needham, MA.

[3] *Electron Notes* What is QAM: quadrature amplitude modulation. (n.d.). Retrieved December 6, 2019, from https://www.electronics-notes.com/articles/radio/modulation/quadrature-amplitude-modulation-what-is-qam-basics.php.

[4] *"QAM Definition."* PC Magazine Encyclopedia, www.pcmag.com/encyclopedia/term/50007/qam.

[5] *"Quadrature Amplitude Modulation (QAM)."* National Instruments Website, National Instruments, Mar. 5AD, 2019, www.ni.com/en-us/innovations/white-papers/06/quadrature-amplitude-modulation–qam-.html.

[6] *Borth, David E., and Wayne Eric Stark.* "Modulation." Encyclopædia Britannica, Encyclopædia Britannica, Inc., 1 Mar. 2019, www.britannica.com/technology/telecommunication/Modulation.

[7] "QAM Formats: 8-QAM, 16-QAM, 32-QAM, 64-QAM, 128-QAM, 256-QAM." Electronics Notes, Incorporating Radio-Electronics.com, www.electronics-notes.com/articles/radio/modulation/quadrature-amplitude-modulation-types-8qam-16qam-32qam-64qam-128qam-256qam.php.

[8] IEEE 802.11ax: The new standard for Wi-Fi. (2018, November 27). Retrieved December 12, 2019, from https://www.gigabit-wireless.com/gigabit-wireless/ieee-802-11ax-new-standard-wi-fi/.

[9] *Huang, D.* (2018, December 1). Wi-Fi 6 fundamentals: What is 1024-QAM? Retrieved December 12, 2019, from https://theruckusroom.ruckuswireless.com/wired-wireless/technologytrends/wi-fi-6-fundamentals-1024-qam/.

[10] Quadrature Amplitude Modulation Explained. (n.d.). Retrieved December 12, 2019, from https://everything.explained.today/Quadrature_amplitude_modulation/.