SEJ QAM

Saarth Mehrotra, Evan Dorsky, Jacob Kingery

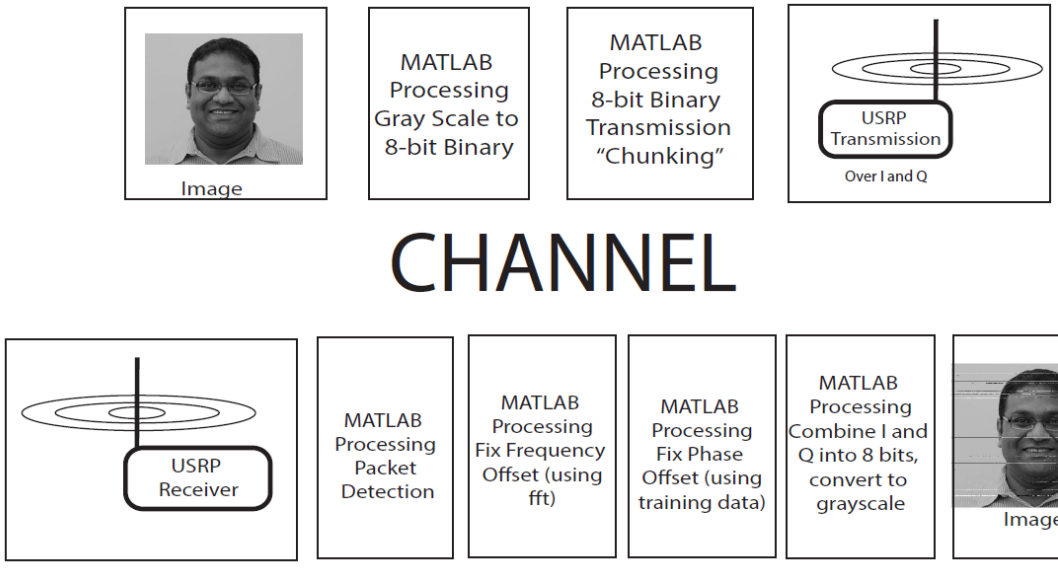
Initial Project Goals:

Our initial goal for this project was to implement wireless 16-QAM transmission and reception using two USRPs. The data we wanted to transmit was a stream of 640x480 8-bit grayscale images. We believed that a phase-locked loop would be necessary to implement in software in order to correct for the ever-present frequency and phase offsets between the transmitter and receiver. We also wanted to look into image compression if we had time.

Revised Project Goals:

After spending quite a bit of time just getting the USRPs to talk to each other and creating an implementation of BPSK, we realized that our goals needed to be revised. We observed that the channel we were dealing with would not allow for 16-QAM due to the SNR being on the low side. We decided that 4-QAM would be our revised goal. We also decided to drop the streaming aspect of the image transmission because of how long it was taking MATLAB to process the received data.

The System:



The image above shows a block diagram of our system. All of the encoding and decoding was done in MATLAB, and the transmission and reception were done via the USRPs. We started with an 8-bit grayscale image and converted each row into a binary string. Sending a single row

per packet, we took every other bit for the I and Q channels and stretched the bits into boxes by duplicating each bit a set number of times (we used 6 bits per data bit). Finally, we added 10 training bits (1s on both the I and Q channels) to the beginning of each packet to be used later to correct phase offset. We sent and received the packets using the provided USRP MATLAB functions.

Once we received all of the data (one packet per row of the image), we had to find the beginning of the signal amongst the noise that immediately preceded the packet. To do this, we found the maximum amplitude of the noise within a window of what we knew was noise and then looked for the point where the amplitude was above some multiple of this. We then used an open-loop FFT method to find the frequency offset between the USRPs and correct for it. Next, we utilized the known training bits to find and correct for the phase offset between the frequency-corrected data and the 4-QAM constellation. Finally, we took our corrected bits, averaged out the duplicates, interlaced the I and Q channels, and converted the binary to the 8-bit integers that are the pixel values.

The Results:

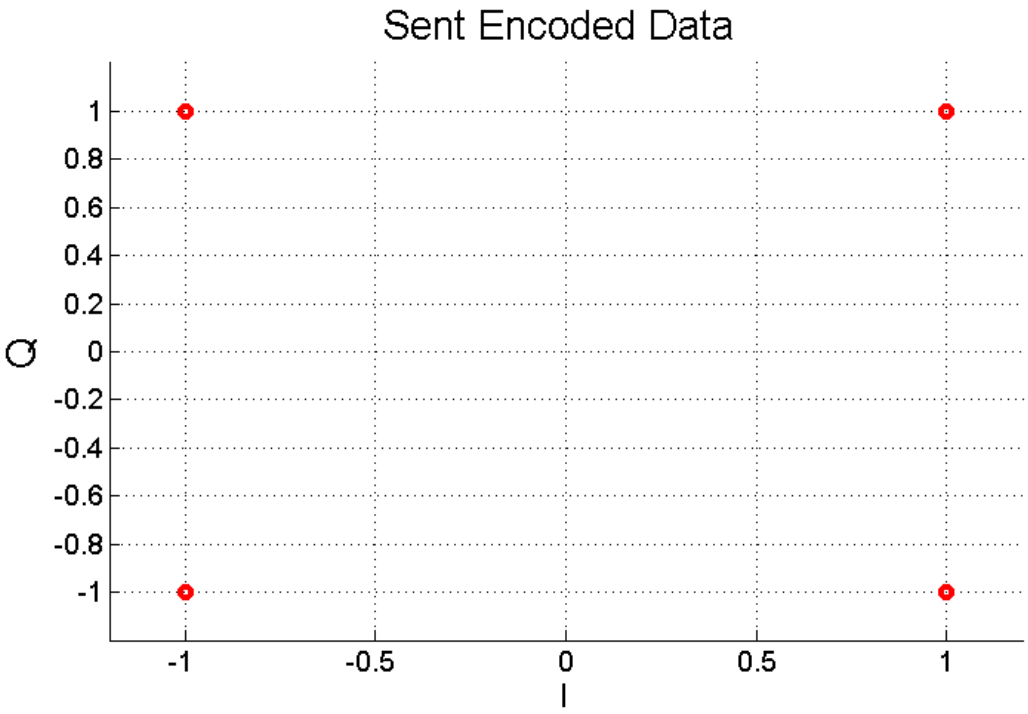


Figure 1. In this graph we see the transmitted data encoded over the I and Q channels as (-1,1), (1,1), (1,-1), (-1-1).

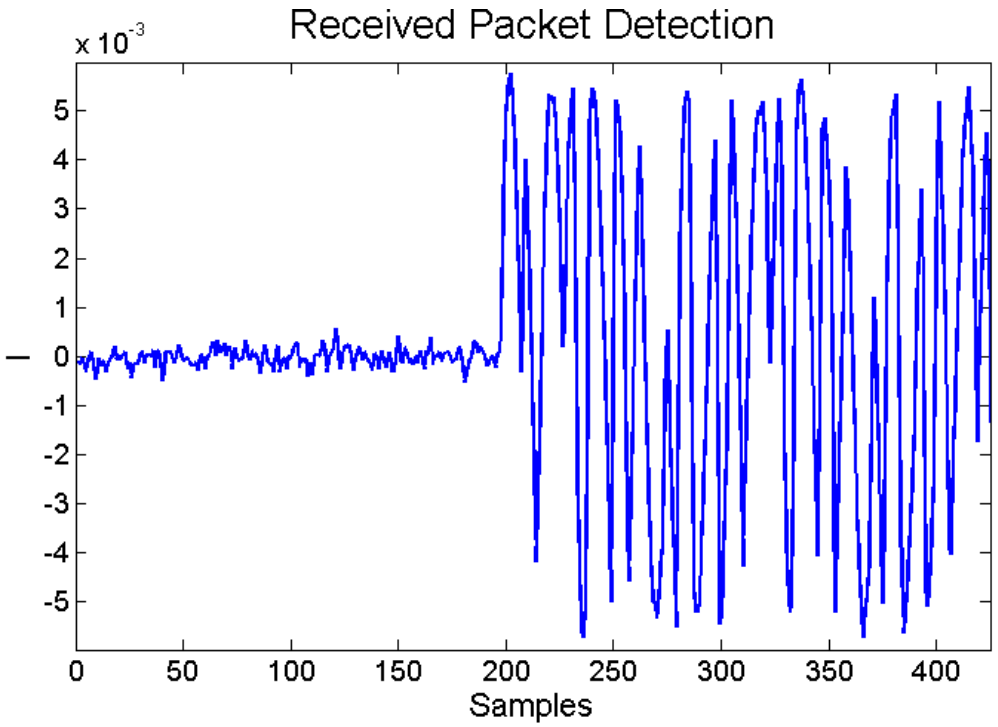


Figure 2. The above graph is an example of received data over the I Channel. Our packet detection would occur around 200 Samples. Everything before that is noise.

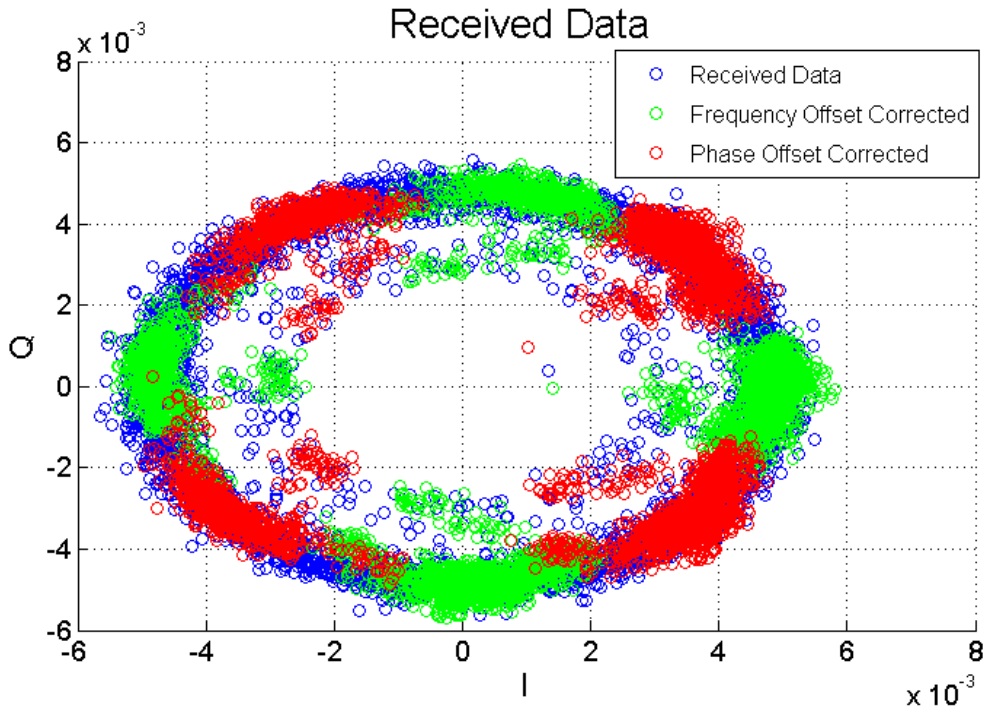


Figure 3. This is a graph of data that we received over the I and Q channels after we have removed the noise from the packet. The green points represent the received data on the complex plane after the open-loop FFT method of frequency offset correction was applied. The red points represent the same data corrected for the phase offset using training data.

The Issues:

As the above figures show, our system was able to successfully transmit the image, but not without errors. The cases where the entire row is incorrect is known as a packet error, and we believe these were caused by the signal detector being off by a certain amount such that all of the decoded bits were offset. The cases where errors begin to show up in the right side of the image are usually caused by the frequency offset not being corrected accurately enough by the open-loop FFT method. This resulted in the drift seen in the above figures that eventually lead to data points crossing the x- or y-axis and being estimated as different bit values. To try to do a better job of correcting the frequency (and phase) offset, we spent a lot of time trying to create an implementation of a Costas loop, discussed in more detail below.

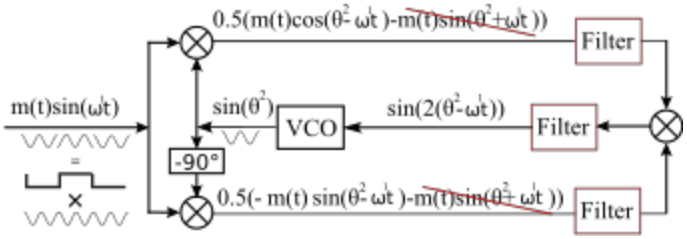
Open Loop Carrier Recovery:

To recover the data from the packets sent between the two USRPs, we had to correct for both frequency (the modulating/demodulating frequencies varied slightly between boxes) and phase (the modulating/demodulating frequencies were not in phase) offsets. To fix the frequency offset of our 4-QAM signal, we raised it to the fourth power and took the FFT. The FFT had a peak corresponding to 4x the frequency offset. We wrote MATLAB code to correct the received data based on the observed peak in the FFT. The results of this step are the green dots in the constellation diagram (Figure 3, above).

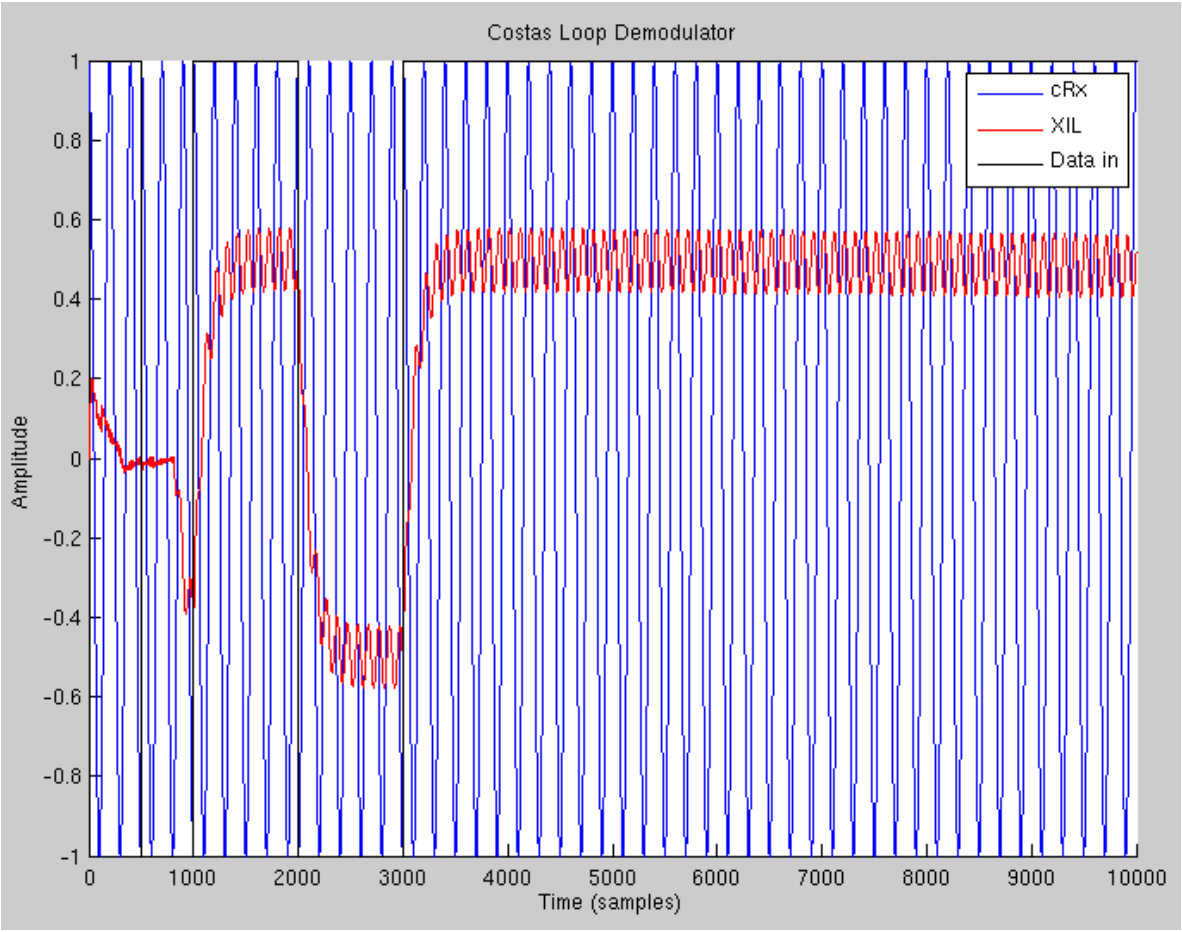
To fix the phase offset of our signal, we used the knowledge that the training bits were all (1,1). We looked at the average angle of the first 10 bits (which should be the training bits) and rotated everything such that this angle lines up with the angle corresponding to 1+1j. The times when this failed were when the first 10 bits were not the training bits (due to our signal detector not finding the correct beginning of the signal), so everything was rotated incorrectly.

Costas Loop:

A Costas loop is a kind of phase locked loop suited for QAM input. It is capable of correcting for both frequency and phase offsets between the carrier and demodulator.



The Costas loop attempts to estimate the carrier wave over time by feeding information about the difference between the actual and estimated carrier back into a VCO. The two multipliers on the left calculate this difference (the low pass filters filter out the upper sidebands), and the multiplier on the right has the effect of squaring the message part of the signal (just BPSK in this case, so either -1 or 1) to remove its effect. Over time, the estimated and actual carrier frequency should align.



Our Costas loop didn't end up working on any of our USRP data, but we did get it working for some test data under certain conditions, as the above plot shows. The blue line is the input, the

black line is the data encoded in it, and the red is the output of the Costas loop representing an estimate of the encoded data. We found that our discrete time implementation of the Costas loop was heavily dependent on the VCO's FFT window size, the sample rate, the actual carrier frequency, and many other parameters. It also only worked on BPSK, so it would not have been able to demodulate the 4-QAM signals that we wanted to send. In the end, we decided to stick to the open loop timing and phase correction, but that didn't stop us from learning about phase locked loops!

Conclusion:

We feel that we learned a lot about creating a real-world communications system (and the difficulties of doing so) while doing this project. However, we also greatly underestimated the difficulty of working with the USRPs, even for implementing a relatively simple system. As such, we ended up spending most of our time trying to get what we thought would be a basic communications system to work at all instead of working on the more interesting goals we had for our project. For the example seen above, the net data rate was 83.12 kb/s, and the bit-error rate was 5.87% (counting packet framing errors -- without packet errors, this would have been significantly lower).

$$\frac{164 \text{ pixels} \cdot 8 \text{ bits}}{164 \text{ pixels} \cdot 48 \frac{\text{bits}}{\text{pixel}} \cdot \frac{1}{2} + 10 \text{ bits}} \cdot \frac{250 \text{ kb}}{\text{sec}} = \frac{83.12 \text{ kb}}{\text{sec}}$$

The above images are the results from our tests, varying the parameters slightly each try. We can clearly see that the most common type of error is a full packet framing error.