Analog and Digital Communications: OFDM Final Project

Lisa Joëlle Hachmann, Anisha Nakagawa, and Paige Pfenninger

February 18, 2018

1 Introduction

Orthogonal Frequency Division Multiplexing (OFDM) is a communication method that is used in most modern wireless communication systems. This method encodes data in frequency using orthogonal frequency carriers, which allows more data to be encoded in a smaller frequency band. This allows for a much higher data rate than most communication systems, in addition to a smaller frequency bandwidth. In our project, we are encoding data with OFDM using MATLAB, sending it through Universal Software Radio Peripherals (USRPs), and then decoding the received signal through another MATLAB script. Our setup is shown in Figure 1.

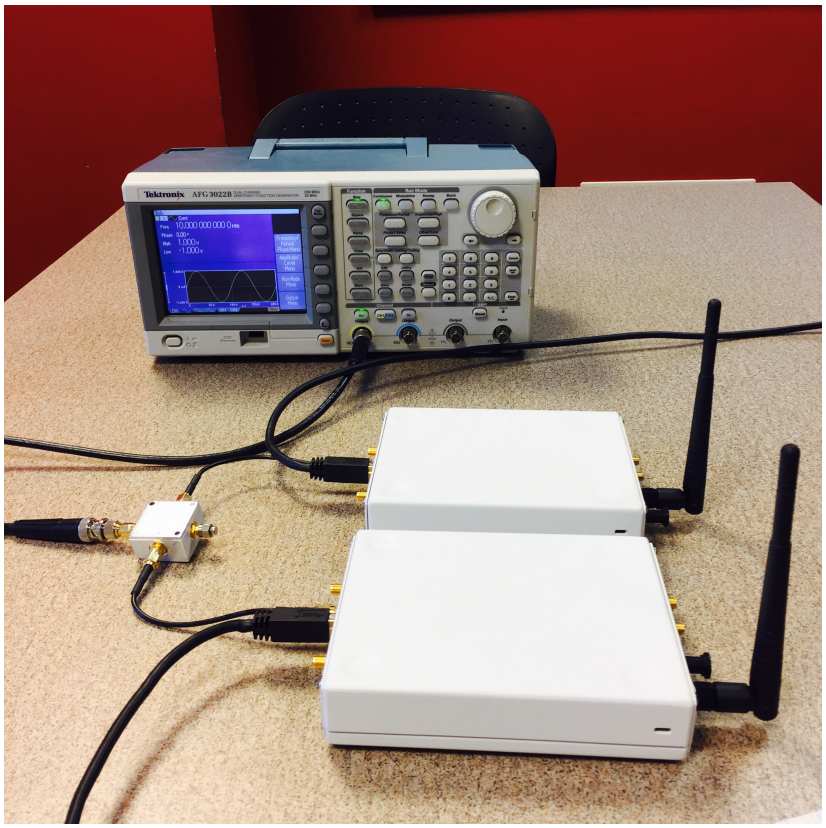

Figure 1: Photograph of our setup using the USRP. The clock signals of the USRPs are synced up using a 10MHz sine wave from the function generator, because correcting for frequency offset is beyond the scope of this project.

Our project is coded entirely in discrete time and frequency, with the conversion to continuous time happening in the USPR radios. However, to fully understand the benefits OFDM it is important to understand how the system works in continuous time.

This system encodes data in frequency as a series of overlapping sincs that are orthogonal: the peak of each sinc lines up with zero crossings of all the other sincs, as shown in Figure 2 on the next page. For this project, we encoded bits in the frequency domain using BPSK, where the bits are grouped together into bins of 64 bits. However, OFDM can also be used with other modulation schemes such as 4-QAM and QPSK. Each bit in the the frequency bin is encoded as it's own sinc in frequency.

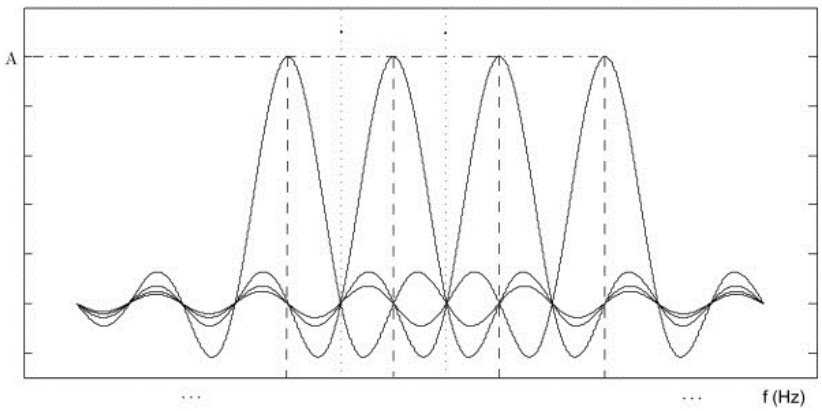

Figure 2: In continuous time, the different frequencies are orthogonal sincs so that the peak of each sinc overlaps with all the other zero crossings. This avoids the need for guard bands in frequency.

For each set of frequency carriers, we take the Inverse Fourier Transform to convert into the time domain, where each sinc corresponds to a signal in time at a different frequency. For each bin of bits, the signals in time are added together and then transmitted. In time, we need to add a cyclic prefix to the beginning of that data section which is just a repeat of the end of that section. This allows for the system to perform circular convolution on each set of data, which is helpful because we can then divide out the channel response. Each packet of bits is transmitted one after another.

The receiver then looks at each package of data at in time, takes the Fourier transform to get into frequency, and decodes the bits from there using a channel estimation.The full block diagram of our system is shown in Figure 3.

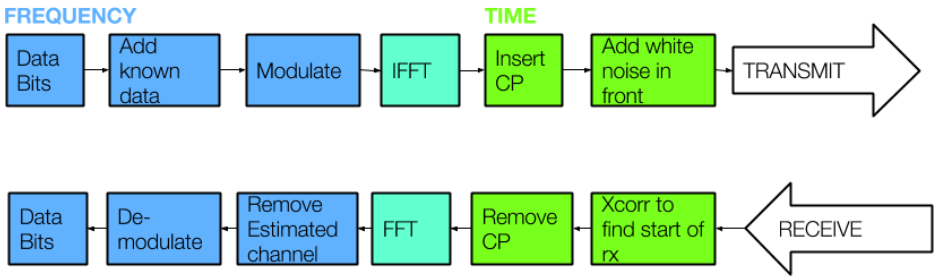

Figure 3: Block Diagram of implemented OFDM scheme

2 Transmit

There are three parts to every transmission: white noise, known data, and data payload.

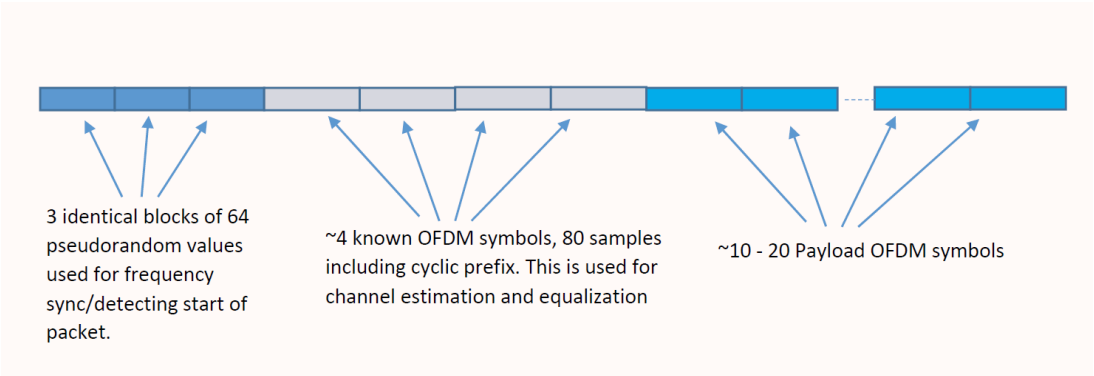

Figure 4: The three parts of an OFDM transmission

The white noise is a known set of data points that is shared between the transmitter and the receiver. It it used by the receiver to find the starting place of the message. It is typically transmitted more than once to increase accuracy. In our transmission system, it is three blocks, with 64 bits per block.

The second part is the known data. This is used by the channel to estimate the data. In our transmission system, it is the length of four 64 bit blocks (or 256 bits). We have 64 unique bits for increased accuracy of channel estimation. In current WiFi systems, these 64 bits are repeated multiple times instead.

The third and final part of our transmission is the actual data that we want to transmit. For this part, the data is broken up into 64 bit blocks, which we send the blocks in groups of five. In between each group of five, we send another four blocks of known data to re-estimate the data. Over longer transmissions, this would help for channel drift.

Our transmission system starts by loading a file containing the bits that we want to transmit along with the 265 bits of known data. We then split the data up into blocks of 64 bits and group those blocks into groups of 5 blocks, having 320 total bits per grouping. We then insert the 256 known data bits in between each of the 5 block groups.

Figure 5: The arrangement of the known data and data payload

Figure 5 shows how the data is arranged. Each blue block is 256 bits of known data, and each green block is five blocks of data where each block has 64 bits.

After combining the known data and the data payload, we translated all of the data points from $[0, 1]$ to $[-1, 1]$. We then organize the data in a matrix such that each column corresponds to one of the 64 sub carrier frequencies. To get the signal into the time domain, we then take the Inverse Fourier Transform of each row.

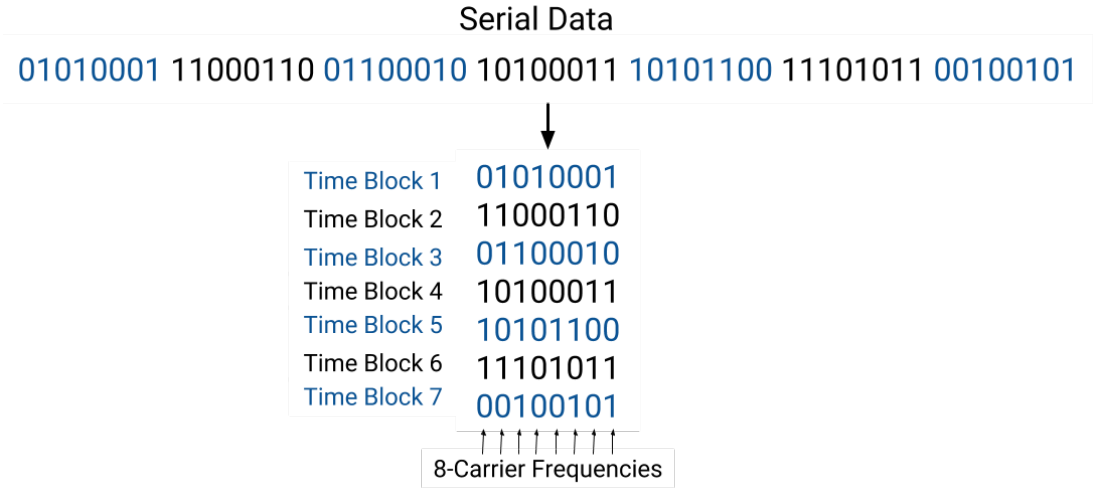

Figure 6: An example of how we arrange the data in a matrix when there are only 8 carrier frequencies.

Figure 6 shows how we arrange the data in the matrix for a simpler system that only has 8 subcarrier frequencies. We take the Fourier Transform of each time block (row) to transform the signal into the time domain. We then add a cyclic prefix to the front of each time block. A cyclic prefix allows us to easily estimate the channel with circular convolution on the receiving end. To insert the cyclic prefix, we take the last 16 columns of our data matrix and copy them to the front of the matrix. This means that we get a matrix with 80 columns (64 subcarrier frequencies, and 16 of the cyclic prefix) and the same number of rows. Research for WiFi systems have found that 16 bit channel estimates well for a 64-bit system. We then convert the signal back into a serial form by concatenating the rows one after another.

We then load in a white noise data file that is known by both the transmitter and receiver. It is 64 bits long. We insert it three times concatenated before the rest of the data. On the receiving side, this will allow us to find the start of the signal.

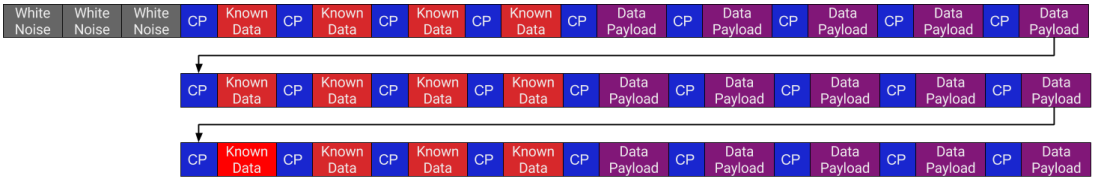

Figure 7: The final data stream before we transmit it

Figure 7 shows what the transmitted data looks like. It starts with the white noise repeated three times, and then we have 3 (or more) groups of known data and data payload. Each group has four 64 bit sections of known data, and each 64 bit section has a 16 bit cyclic prefix. The known data is then followed by five blocks of 64 bits of data payload and each block also has a 16 bit cyclic prefix.

This data stream is then converted into a file that is readable by the USRP and sent to the receiver. We transmitted with a gain of 70dB at 2.491MHz with a sampling rate of $750 \times 10^3 \frac{bits}{second}$.

3 Receive

On the receiver, we read data into MATLAB and begin processing it by finding the starting point of the transmission. We know the transmission consists of three blocks of white noise, known data, and then the data payload, with repeating known data and data payloads as designed. The starting point is found by cross correlating the received transmission with the known white noise we include at the start of the data. Originally, we were picking up a glitch from the hardware, but cross correlating with three blocks of known white noise removed this issue. After the white noise, we briefly move the serial information into a matrix to see time domain signals in blocks of 80 (the 64 bit data blocks with 16 bit cyclic prefixes) to see the subcarrier frequencies as in Figure 6 on the preceding page We strip the cyclic prefix by removing the first four columns of the data. Then, we take the Fast Fourier Transform of the received known data. In the frequency domain now, we estimate the channel by dividing the received version of the known data with the known data that we know the transmitter sent. Of course, the received version is garbled by the channel, but this gives us 4 blocks of 64 to estimate the channel over. We take the average for each block and then average with all four repititions to get a good estimation.

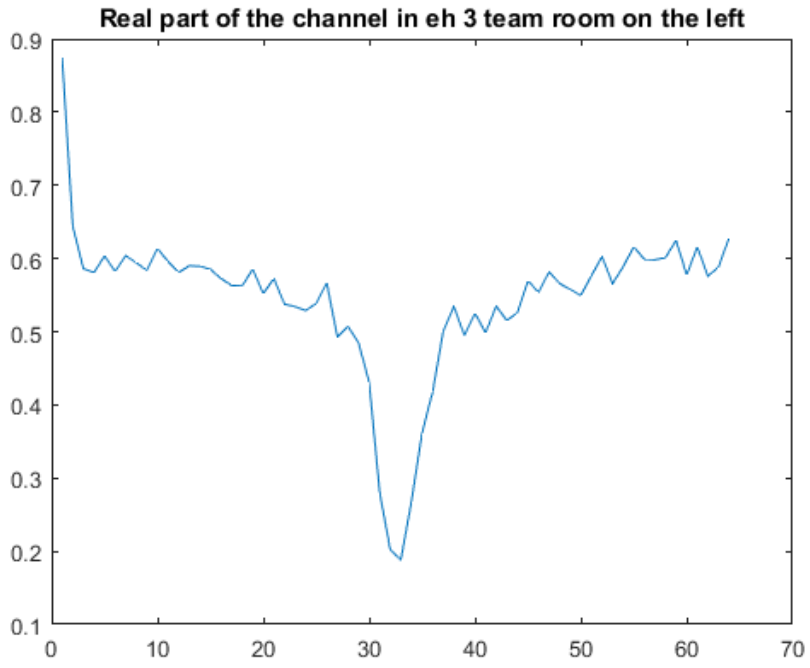

Figure 8: Averaged channel in an East Hall team room

Figure 8 shows the channel in a team room in East Hall. This channel was consistent within team rooms, but was

largely garbled elsewhere. We noticed that the 30-35 indices of the channel were often times much worse than the rest of the channel. One way we considered improving the channel would be to not transmit anything in those bins, as they're likely to have an error.

We now take the Fast Fourier Transform of each block of the data payload and remove the effects of the channel by dividing out the averaged channel at the same bin. Now we have our estimated data payload and we move back into serial. To estimate, we look to see if there is a sign change between each of the bits, and if so, we change from a 0 to a 1 or back. We compare the first bit of data, knowing that it should be a 0, and then make sure the sign changing estimation is oriented with that. If the first bit has an error, then our entire string will be estimated wrong and we'll have the inverse bit error rate. This method can be improved in future work.

Now having estimated bits, we can convert it to a string. Because we happen to know our transmitted data, we can compare the estimated bits with the original bits and find our bit error rate.

4 Results

We successfully achieved a bit error rate of 10^{-3}. During this transmission, we were transmitting a total of 2352 bits at a sample rate of $750 \times 10^3 \frac{bits}{second}$, so the transmission took 0.0032 seconds. Our data payload for that transmission was 960 bits, which gives us a final data transmission rate of $3 \times 10^5 \frac{bits}{second}$.

Unfortunately, our system is very sensitive and only likes to work in particular spaces where we can get a good channel response as seen in figure 8 on the preceding page. In future iterations, we would like to be able to build a more robust transmission system.

All of our code and the necessary README can be found at: `https://github.com/ppfenninger/OFDM/`.

4.1 Reflection

We found a large range in success based on our location and the channel. Even within the same room, we would have bad transmissions for a bit and then an amazing one. While we can't explain why the channel would change so frequently, we can definitely corroborate that when the bit error rate was bad, the channel seemed awful and hard to estimate even with four blocks of known data.

5 Future Work

In order to improve our transmission, we would consider lengthening our transmitted data and not transmitting over certain parts of the channel. In order to lengthen our transmitted data, we think we'd need to have more channel estimation between data blocks to account for channel drifting. We did consistently see the middle portion of the channel behave badly, and know our bit error rate would be better if we did not transmit in those subcarrier frequencies, similar to how WiFi does not transmit at some subcarriers.

In the future, we would also want our radios to transmit when the clock signals are not synced up to a reference clock signal. However, this would include extensive work in correcting for the frequency offset, which was beyond the scope of this project.

References

[1] Siddhartan Govindasamy. 2017.

[2] Dušan Matic *Mathematical Description of OFDM.*
http://www.wirelesscommunication.nl/reference/chaptr05/ofdm/ofdmmath.htm

[3] Principles of Wireless Communication course, Olin College, Lab 3 handout. April 2016