Wireless receivers: Algorithms and Architectures
OFDM project

Thomas Lenges, Renuka Singh Virk

December 30, 2024

1 Introduction

This project focuses on the implementation of an Orthogonal Frequency Division Multiplexing (OFDM) system designed to transmit and receive acoustic data through a pair of speakers and a microphone. Building on the structure of assignment 7, which explored single-carrier audio transmission, this project extends the concepts to the more advanced OFDM technique, adapting them for multi-carrier transmission.

OFDM, known for its resilience to multipath effects and efficient use of bandwidth, is well-suited for communication in challenging acoustic environments. The implemented system features a BPSK-mapped single carrier preamble, a BPSK-mapped OFDM training symbol for synchronization, and QPSK-mapped OFDM data symbols with cyclic prefixes to mitigate inter-symbol interference, and advanced techniques such as channel equalization and phase tracking to enhance robustness.

Performance evaluation under various channel conditions demonstrates the system's capability for effective error correction and reliable data recovery. A spectral efficiency analysis further highlights the trade-offs introduced by cyclic prefix overhead, offering insights into optimizing system parameters.

Finally, extensions such as image transmission showcase the versatility and potential applications of the system.

2 Implementation

2.1 Transmitter

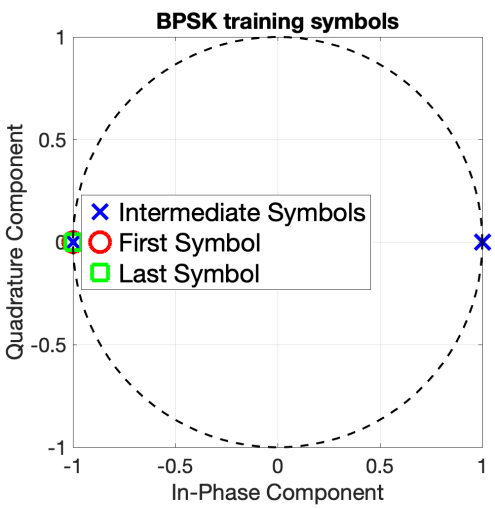
The transmitter file `tx.m` essentially receives a series of bits and returns a signal `txsignal` to be sent to the receiver as described in the following subsections.

2.1.1 Preamble

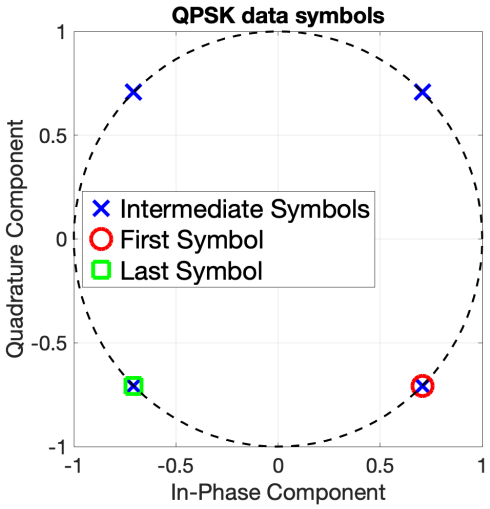
In order for the receiver to determine the location of the data, we use a preamble generated using the linear-feedback shift register (LFSR) approach. We thus generate a pseudo-random sequence of bits. The bits are then mapped to BPSK symbols, oversampled, and pulse-shaped with a root-raised-cosine filter. This part of the project does not change from assignment 7.

2.1.2 OFDM Training symbols

Because each subcarrier presents a different phase error and different channel conditions, we need a reference OFDM symbol to correctly recover the data. A training symbol is thus generated using LFSR and mapped to BPSK, as was done for the preamble. However, this symbol must have the same size as our data OFDM symbols and is thus mapped on $N_{subcarriers} = 256$ subcarriers.
We can visualize in Figure 1a the result of the BPSK-mapping of the training symbols.



(a) BPSK-mapped training symbols



(b) QPSK-mapped data symbols

Figure 1: Training and Data Symbols

2.1.3 OFDM data symbols

The actual data to be transmitted is mapped using QPSK and then divided into packets of $N_{subcarriers}$ parallel streams. Each stream is fed to the `osifft.m` file to render output samples in the time domain. The last 50% of these samples are then copied and prepended to the front of the samples, forming a **cyclic prefix**. The cyclic prefix ensures that the convolution of the transmitted signal with the channel impulse response becomes a circular convolution, which is necessary because the Discrete Fourier Transform (DFT) assumes circular convolution when transforming signals. Without the cyclic prefix, leaving an empty guard interval would result in linear convolution, causing distortions and a loss of orthogonality among subcarriers.
We can again visualize that the OFDM data symbols are correctly mapped to QPSK (Figure 1b).

2.1.4 Final Transmit Signal

Once we have obtained our preamble (in the time domain), the OFDM training symbol, and the OFDM data symbols (both also in the time domain), we normalize each of these individually to achieve unit power and concatenate them. We then upconvert the signal and ensure that it is real, to form the final signal `txsignal` to be sent over to the receiver (Fig. 2a).

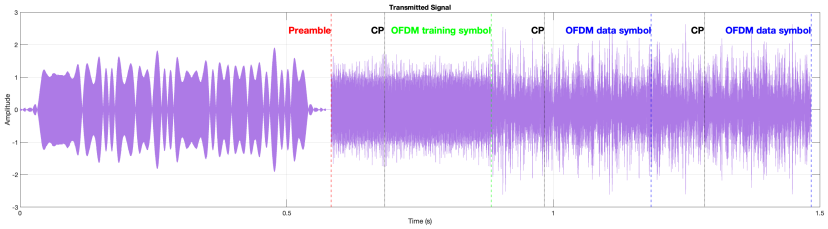
We can also visualize the spectrum of the transmitted signal to ensure a correct modulation. As we see in Figure 2b, the signal is centered around $-f_c = -4$kHz and $f_c = 4$kHz as expected.

The very narrow peaks at -4kHz and 4kHz in the spectrum are likely caused by the pulse shaping applied to the preamble signal. With a pulse length of 2000, the preamble signal exhibits a relatively consistent amplitude over a longer duration, resulting in a spectrum with narrow, prominent frequency components at -4kHz and 4kHz (the carrier frequencies).

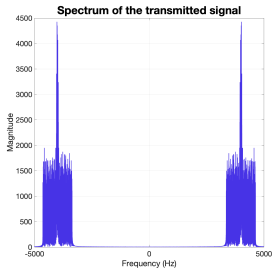
The longer pulse length reduces the bandwidth of the preamble signal, concentrating its energy around the carrier frequencies. Additionally, the smooth nature of the time-domain preamble signal (compared to the more dynamic OFDM symbols) contributes to the sharper spectral peaks observed. These peaks are less pronounced in the spectrum of the OFDM symbols, which exhibit higher frequency variations due to their complex structure.

2.2 Data transmission

The `audiotrans_ofdm.m` file is an adaptation of the file `audiotrans.m` from assignment 7. We define new constants such as the number of subcarrier frequencies (here 256), the spacing frequency (5Hz), and the oversampling factor.



(a) Transmitted signal (time domain).



(b) Transmitted signal (frequency domain).

Figure 2: Transmitted signal in time and frequency domains.

2.3 Receiver

2.3.1 Signal downconversion

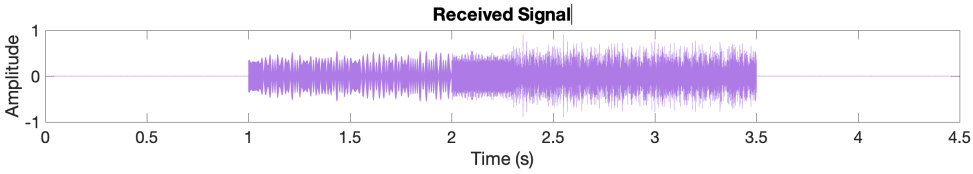
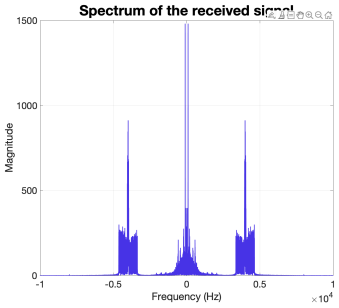


Figure 3: Received signal (time domain).

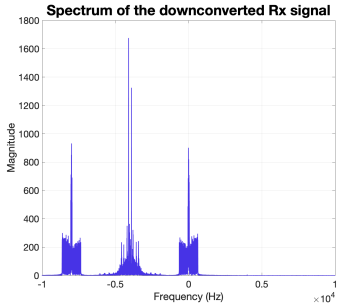
We can start by visualizing the received signal and notice that the padding applied in the `audiotrans_ofdm.m` file adds $f_{sampling}$ (`conf.f_s`) on both sides of the signal (cf. figure 3).

This padding is applied to ensure that during the recording process, we do not miss the beginning or end of the data.

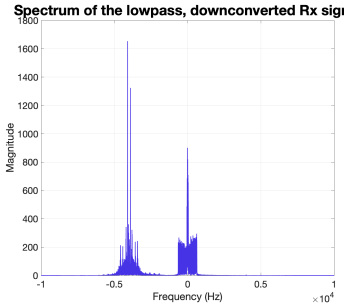
The received signal needs to be downconverted and passed through a lowpass filter to remove the undesired image as is showed in Figure 4.



(a) Spectrum of the raw received signal.



(b) Received signal spectrum after downconversion.



(c) Final signal spectrum after lowpass filtering.

Figure 4: Received signal downconversion and lowpass.

2.3.2 Preamble detection

Once the received signal has been shifted back around DC and passed through the lowpass, we use the `find_preamble.m` function to identify the start of the data. This function is similar to the `framesync.m` function used throughout the semester. It computes the correlation between the regenerated preamble and the received signal as was done in the labs. If the correlation exceeds a certain threshold, the corresponding index is outputted as the beginning of the data transmission. The threshold value is set manually by looking at the correlation plot (Figure 6)

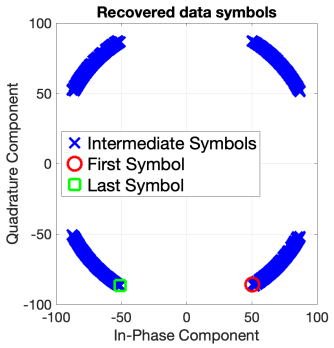
2.3.3 Perfect channel reception

For a short transmission under ideal conditions, no particular processing of the received data would be necessary.

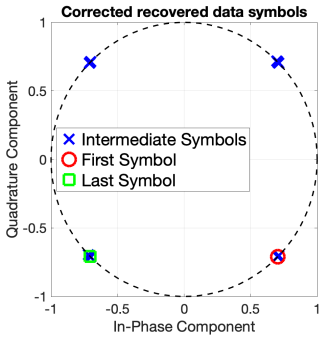
However, as visualized in Figure 5a, even for ideal channel conditions, the symbols are slightly shifted. For longer data transmissions, this would lead to some received symbols being in the incorrect quadrant. To handle this, we start by comparing each received training symbol with the transmitted symbol, estimate the phase, and rotate the symbols back to recover the data. As can be seen in Figure 5b, this intuitive method works quite well. Although this method is the first one we tried, we also implemented the classical channel equalization presented in the project instructions and kept this method, since it is an extension of our naive method. Please note that the channel equalization implemented corrects only the amplitude, as the phase tracking is later implemented and handles the phase correction.

We use the training symbol to estimate for each subcarrier the channel impulse response H, and use the estimate to correct the symbols as follows:

$$a_{estimate}[n,m] = \frac{z[n,m]}{H[n,m]}$$



(a) Raw received symbols using bypass mode.



(b) Corrected received symbols using bypass mode.

Figure 5: Raw and corrected received symbols using bypass mode.

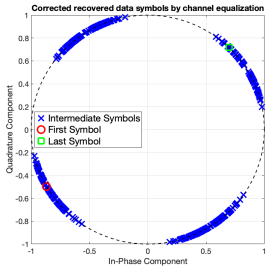
where $z[n,m]$ are the received symbols in the frequency domain, $a_{estimate}[n,m]$ are the estimated transmitted symbols, and $H[n,m]$ is the channel impulse response.

Both the naive phase correction and the channel equalization methods are coded in separate files (`naive_phase_correction.m` and `channel_equalization.m`) and called in the `rx.m` file. The synchronization method can be selected in the `audiotrans_ofdm.m` file by setting `conf.synchronization` to the desired option. For more details, please refer to the code comments.

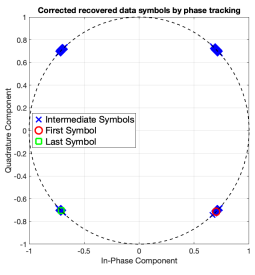
Figure 6: Test statistic for preamble detection.

2.3.4 Imperfect channel reception

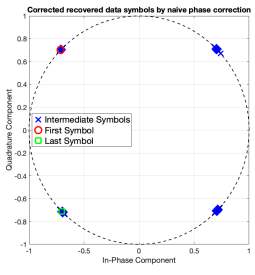
If we switch to the matlab mode, we can observe the performance of the different methods tested in Figure 7. It makes sense that for QPSK, the most important aspect of the symbols is their phase, and so without any correction of the phase, the bit error rate is quite high, as can be seen in Table 7d. 1024 bits were transmitted in the three cases presented in Figure 7, and the BER indicates that naive phase correction is sufficient for that amount of bits. Furthermore, some form of phase correction, as simple as it may be, is necessary.
The details of the implementation of continuous phase tracking are discussed in section 5.1.1.



(a) Channel equalization without phase correction.



(b) Channel equalization and phase tracking.



(c) Naive phase correction, no channel equalization.

Method	BER %
Channel equalization	0.5
Channel equalization with phase tracking	0
Naive phase correction	0

(d) Bit error rates for different methods.

Figure 7: Comparison of different processing methods at the receiver.

2.3.5 Symbols demapping

The final step implemented by the receiver is to return a bitstream. This is done via the `demapper.m` function, which assumes that the bits are mapped using Gray mapping scheme for QPSK.

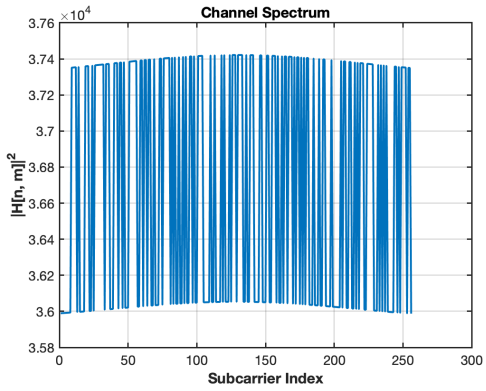
3 Channel analysis

3.1 Channel spectrum

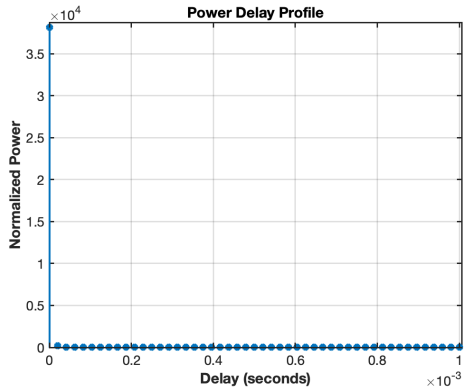
The lowpass filter used in the `rx.m` file is not ideal, as real-world filters deviate from theoretical brick-wall filters. Instead of perfectly passing all frequencies within the desired band while completely attenuating those outside, practical filters have a transition band and non-uniform attenuation across their passband. This imperfection causes the edge subcarriers to experience more attenuation compared to the middle subcarriers, which lie well within the filter's flatter passband. As a result, the spectrum of the channels shows higher magnitudes for subcarriers near the middle indices. This effect is witnessed in Figure 8a, where the edges are indeed attenuated.
Now focusing on Figure 8c, it is plain to see that each subcarrier is attenuated differently under real channel conditions.
The peaks translate to a higher magnitude of the channel, which can be confirmed by the channel magnitude evolution over time plots, where if we compare Figures 9a and 9b, the red ellipses highlight these correlations. (cf. section 3.3).

3.2 Power delay profile

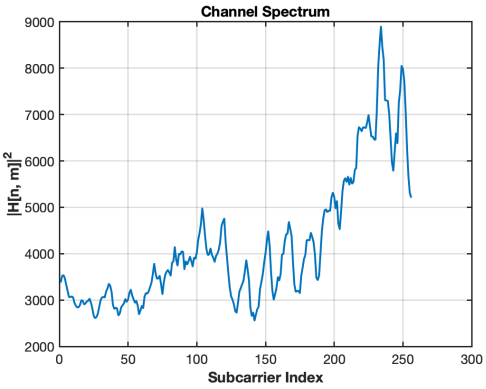
The Power Delay Profile (PDP) is a representation of how the power of a transmitted signal is distributed over different time delays due to multipath propagation. The PDP describes the time dispersion of the channel which can explain inter-symbol interference (ISI). A perfect channel has a Power Delay Profile (PDP) with a single peak at zero delay, since there are no delays and thus all of the signal power is concentrated around a time delay of zero (Figure 8b).
For realistic non-perfect channel conditions (matlab mode), the PDP now shows some of the signal power contained around small delays. This is due to multipath propagation, where the transmitted



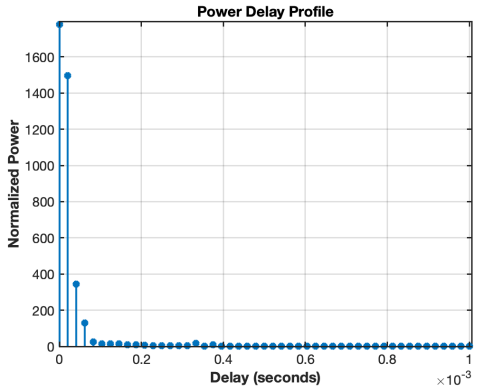
(a) Channel spectrum with bypass mode.



(b) Power delay profile for bypass mode.



(c) Channel spectrum with matlab mode.



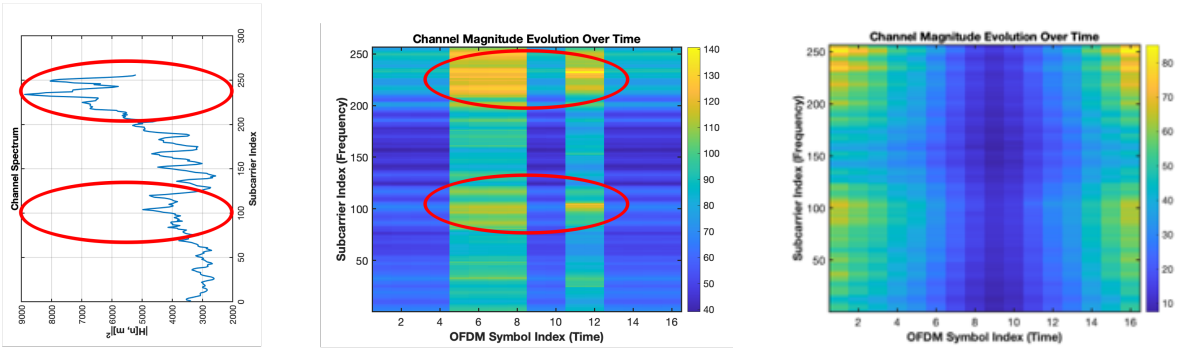
(d) Power delay profile for matlab mode.

Figure 8: Channel spectrum magnitude and Power Delay Profile comparison of bypass (ideal channel) and matlab (real-life channel) modes.

signal takes multiple paths to reach the receiver, resulting in the observed delays. These delays occur because the signal reflects off various surfaces or obstacles, causing some paths to be longer than others, which spreads the signal power in the PDP (Figure 8d).
The more obstacles there are between the transmitter and the receiver, the more peaks we see at different delays. To visualize this, we delibarately created multipath propagation as is discussed in section 5.2.2.

3.3 Magnitude over time

If the channel magnitude remains relatively constant over time, the system can rely on a single channel estimate derived from the training symbol for equalization.
However, if the channel magnitude changes significantly over time, relying on a static channel estimate can lead to errors. In such cases, tracking the channel evolution dynamically becomes essential and requires more frequent channel estimation.
The magnitude variation over time can be observed in Figure 9c, where we start the transmission at maximum volume, then slowly decrease the volume down to zero and progressively increase it back to the maximum. The effect is mirrored on the plot, where the beginning and end of the transmission show bright colors (indicating high magnitude), and the middle of the transmission shows dark blue, synonym of a zero volume.



(a) Constant volume. (b) Constant volume. (c) Varying volume.

Figure 9: Channel evolution over time under different conditions.

4 Spectral Efficiency Analysis

Spectral efficiency measures how efficiently the available spectrum is used to transmit data. Higher spectral efficiency translates to more data being transmitted over a given bandwidth without increasing the occupied spectrum.
The general formula for spectral efficiency computation is the data rate (in bps) divided by the bandwidth (in Hz):

$$\text{Spectral efficiency} = \frac{\text{data rate [bps]}}{\text{Bandwidth [Hz]}} \tag{1}$$

4.1 Cyclic Prefix impact on spectral efficiency

The cyclic prefix (CP) is necessary to avoid inter-symbol interference. However, this comes at a cost; adding a cyclic prefix reduces the spectral efficiency. Indeed, the CP increases the total duration of each OFDM symbol (hence decreasing the data rate), yet it does not add any additional data. In the following sections, we analyze the impact of CP on the spectral efficiency.

4.1.1 CP Overhead (β)

The cyclic prefix overhead is the fraction of the OFDM symbol duration that is taken up by the cyclic prefix:

$$\beta = \frac{T_{\text{CP}}}{T_{\text{CP}} + T_{\text{symbol}}} \tag{2}$$

where:
- T_{CP} is the duration of the cyclic prefix in seconds.
- T_{symbol} is the duration of one OFDM symbol.

We start the project with a cyclic prefix of 50% of the OFDM symbol duration:

$$T_{\text{CP}} = \frac{T_{\text{symbol}}}{2} \tag{3}$$

Thus, the CP overhead is:

$$\beta = \frac{\frac{T_{\text{symbol}}}{2}}{T_{\text{symbol}} + \frac{T_{\text{symbol}}}{2}} = \frac{1}{3} \tag{4}$$

4.1.2 Spectral Efficiency with Current CP

The spectral efficiency with the current CP overhead is given by:

$$\eta_{\text{current}} = M \times (1 - \beta) \tag{5}$$

where:
- M is the modulation order (2 in our case (QPSK)).
With our initial values, we have a spectral efficiency of

$$\eta_{\text{current}} = 2 \times (1 - \frac{1}{3}) = 1.33 \text{ bits/s/Hz} \tag{6}$$

4.1.3 Spectral Efficiency with Reduced CP

The initial choice to set the CP duration to 50% of the duration of the OFDM symbol is arbitrary. To improve spectral efficiency, we can reduce the CP duration.
However, it is crucial to ensure that the CP is long enough to prevent inter-symbol interference. This is achieved by setting the CP duration to match the measured delay spread of the channel.

With this reduced CP duration, the spectral efficiency becomes:

$$\eta_{\text{reduced}} = M \times (1 - \beta_{\text{reduced}}) \tag{7}$$

4.1.4 Spectral Efficiency Gain

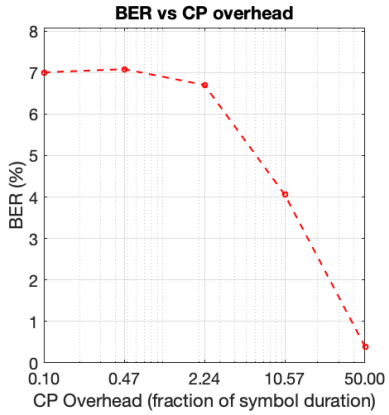
The gain in spectral efficiency from reducing the CP is:

$$\text{Spectral Efficiency Gain} = \left(\frac{\eta_{\text{reduced}} - \eta_{\text{current}}}{\eta_{\text{current}}} \right) \times 100 \tag{8}$$

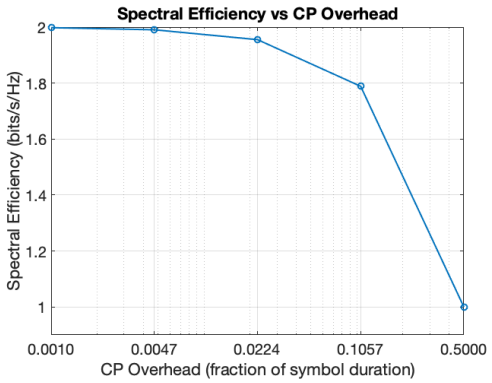
4.2 Practical Analysis of Spectral Efficiency Using Measured Delay Spread

In the function `channel_equalization.m`, we call the function `spectral_efficiency_analysis`, which computes the spectral efficiency with the given CP as well as the improved spectral efficiency by choosing the CP duration as the measured delay spread.
The delay spread values are around $1 \cdot 10^{-3}$[s]. This leads to a reduced CP overhead of 0.5% versus the arbitrary 33.33%, and thus the new spectral efficiency is 1.99bits/s/Hz which leads to a spectral efficiency gain of 49.24%.

As shown in Figure 10, a larger cyclic prefix overhead leads to a smaller bit error rate as it decreases ISI, however this leads to a reduced spectral efficiency. There is thus a tradeoff to be found. From Figure 10a, the CP overhead should be somewhere between 10 and 50%. We thus ran the experiments again with more values within this range, the results are shown in Figure 11. A good bit error rate comes at the cost of a reduced spectral efficiency, it is thus a value to adapt to the desired application.

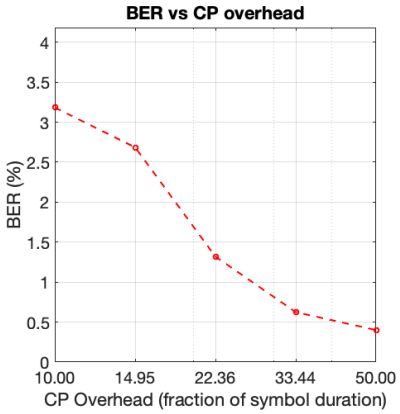


(a) BER vs. cyclic prefix overhead

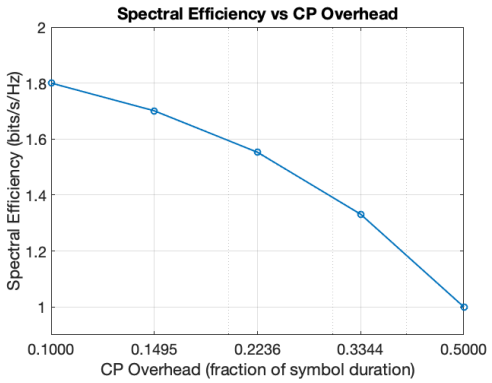


(b) Spectral efficiency vs. cyclic prefix overhead

Figure 10: Impact of cyclic prefix overhead on BER and spectral efficiency (5 frames for each CP overhead).



(a) BER vs. cyclic prefix overhead



(b) Spectral efficiency vs. cyclic prefix overhead

Figure 11: Impact of cyclic prefix overhead on BER and spectral efficiency (5 frames for each CP overhead).

5 System extension

Now that our simple OFDM system works well for relatively simple transmission of random bitstreams, we can push our system to its limits and study what areas leave room for improvement.

5.1 Transmitting image data

Our system works for random bitstreams, there is no reason why it would not work for bitstreams of other types of information.
We choose here to transmit a hand-made image of a smiley. The image is 32px by 32px and each px has a depth of 8 bits with 8 `'0'`s for black and 8 `'1'`s for white (cf. figure 12). Using only black and white makes it easier to see the quality of the transmission, as any gray in the image indicates errors in the transmission of the bits.

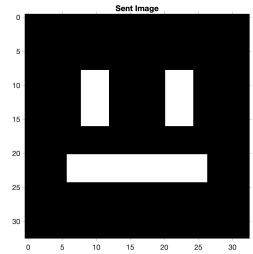
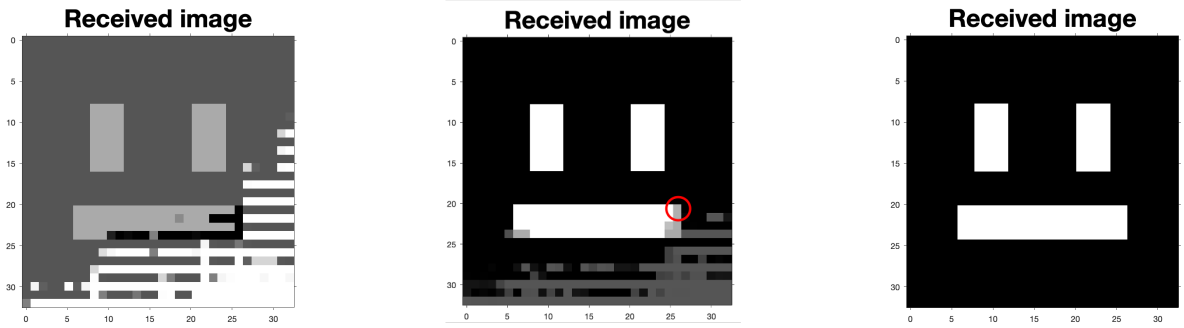


Figure 12: Smiley image used for transmission.

5.1.1 Continuous phase tracking

Up until now, we have only transmitted rather short bitstreams, where a naive phase correction was sufficient.
However, for longer transmissions, this method is no longer efficient. In fact, with image transmission, we can visualize and calculate more or less from which bit naive correction starts failing us. This is illustrated in Figure 13, where it is striking that the transmission of the image failed in Figure 13a. Indeed, the image should contain only black and white pixels, yet most of the image is in shades of gray. Although the transmission failed, the overall shape of the smiley is still visible. This occurs because binary images effectively use only two QPSK symbols, corresponding to black and white pixels. In this case, they are diagonal QPSK symbols (either 1+j and -1-j or -1+j and 1-j). Thus, even if the received symbols are no longer in their original quadrants due to rotation or noise, the initial 180-degree phase difference between the symbols representing black and white pixels is likely to remain unchanged. This ensures that the contrast is preserved, making a transition from black to gray more probable than a transition from black to white.
Figure 13b presents a red circle highlighting the first bit where naive phase correction is no longer sufficient. It corresponds to around the 5320^{th} bit. Thus from this number of bits on, continuous phase tracking is necessary for proper data recovery.
Finally, Figure 13c shows that our implementation of the channel equalization with phase tracking allows for a perfect reception of the image under good conditions (no obstacles created, constant volume, reduced ambient noise).
The continuous phase tracking we implement is the generalization of the Viterbi-Viterbi algorithm

studied in the course, now applied independently to each subcarrier frequency. Indeed in OFDM, each subcarrier presents different channel conditions and phase deviations, which makes it impossible to use a single estimation for all subcarriers. The code for the continuous phase tracking is in the `phase_estimator.m` file, and then called for each subcarrier in the `phase_tracking.m` file.



(a) No channel equalization or phase correction.

(b) Naive phase correction with symbol normalization.

(c) Channel equalization, continuous phase tracking.

Figure 13: Received image for different processing methods.

5.2 Varying channel conditions

5.2.1 Impact of volume on transmission

Up until now, we have transmitted the audio signal at maximum volume, while always checking that there was no clipping of the signal.
We can thus wonder how robust our system is to low volume transmission, or changing volume transmission. In the following section, we use the term 'low volume' to refer to the minimum possible sound intensity of the MacBook Pro (2.3 GHz Quad-Core Intel Core i7) while maintaining the maximum intensity on the provided speakers.
For 5 frames, each time transmitting the same image, we obtained an average bit error rate of 4.26% for low volume transmission. For reference, at maximum volume, we obtained a BER of 0.42%.
We also transmitted the image while varying the volume as explained in section 3.3. The resulting image is displayed in Figure 14b, where we can clearly see for which bits the volume was decreased.
These two experiments highlight that our system is not only not robust to volume variations, but additionally requires specific volume conditions to work properly.

5.2.2 Creating multipath propagation

If we add physical obstacles between the transmitter and the receiver, we can notice a change in the channel and a direct impact on the quality of the received signal.
This is illustrated in Figure 14c, where the received image, even after channel equalization and continuous phase tracking correction, presents a lot of gray pixels, indicating falsy transmission. The right plot illustrates the power delay profile (PDP), which, as expected, shows that some of the signal power is contained at delays other than zero. To ensure that our experiment was indeed highlighting multipath propagation effects and not e.g. lower volume due to signal attenuation by some of the obstacles, we made sure to compare the signal amplitudes for low volume transmission and multipath propagation transmission. In the case where we artificially create obstacles, the amplitude of the received signal is about ten times that of the low volume transmission, confirming that the observed effect is indeed due to the obstacles and not a reduced volume.
The obstacles are simply two boards, which are positioned around the speakers in such a way as to maximize the reflection of the transmitted signal off their surfaces.
The resulting BER was 8.72% for transmission of the image for 5 frames.

5.2.3 Background noise

The last experiment we did in order to change the channel conditions was to play a 4kHz background noise throughout the transmission of the image for 5 frames. This led to a BER of 8.4%.
The different bit error rates of the experiments described in sections 5.2.1, 5.2.2 and 5.2.3, as well as for good channel conditions are summarized in Figure 14d.

6 System limitations

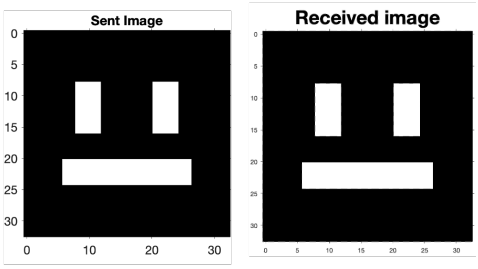
As presented already in the previous sections, channel conditions have a great impact on the quality of the transmission. We have seen that a low or varying volume, as well as adding obstacles in the path have a great impact on the BER.
There are thus some improvements to be made in order to render our OFDM system robust to varying channel conditions.
Additionally, the sound generated during data transmission is high-pitched and potentially uncomfortable for real-world applications. This is another area we could lean into if we still want to transmit information through audio.
The bit error rate of our system could be improved quite a bit by using e.g. multiple input multiple output (MIMO) to exploit spatial diversity. We could also consider using 16-QAM instead of QPSK to improve our transmission data rate.

Finally, throughout this project, we kept the initial value of 256 subcarriers; running the code with different values leads to similar results, but deeper investigation could be conducted to study the impact of the number of subcarriers on the overall quality of the system.

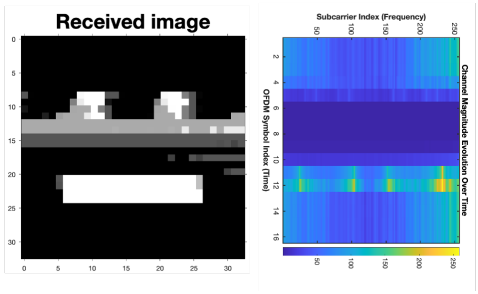
7 Conclusion

In conclusion, our OFDM system successfully transmitted data with acceptable bit error rates under favorable channel conditions.

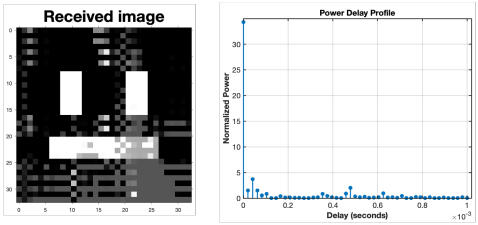
One of the primary challenges was handling phase variation. Continuous phase tracking using the Viterbi-Viterbi algorithm significantly improved system performance. However, for short data transmissions (below approximately 5230 bits), simple phase correction proved to be sufficient.



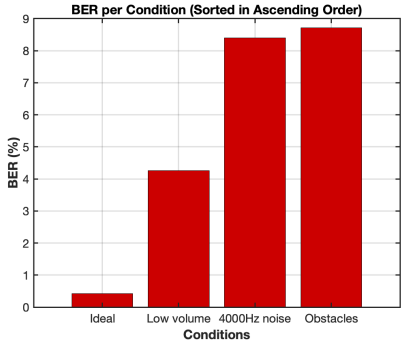
(a) Ideal channel conditions.



(b) Varying the volume.



(c) Multipath propagation.



(d) Histogram of the bit error rates under different channel conditions (in %).

Figure 14: Image transmission for different channel conditions.

We also addressed challenges like channel equalization, though it did not yield noticeable improvements in system performance. This is likely because QPSK primarily relies on phase information, making amplitude correction less impactful, but, as mentioned, it would be key for a system using QAM mapping.

When tested in public areas, our system performed well under moderate channel conditions.

However, it struggled in the presence of loud ambient noises (artificially generated during our tests) and volume variations. Additionally, we noticed limitations in the system's robustness when the

speakers or microphone were in motion, which is worth investigating to further improve the quality of our system.

Despite these limitations, our OFDM transmitter and receiver represent a strong foundation for further development. With additional improvements, such as enhanced noise resilience and better handling of dynamic channel conditions, this system could be adapted for more robust and versatile applications.