OFDM
Laboratory Session

Part 1 – Symbols and Multipath

Symbol

A symbol may be described as either a pulse in digital baseband transmission or a tone in passband transmission using modems. A symbol is a waveform, a state, or a significant condition of the communication channel that *persists*, for a fixed period of time. A sending device places symbols on the channel at a fixed and known symbol rate, and the receiving device has the job of detecting the sequence of symbols in order to reconstruct the transmitted data. There may be a direct correspondence between a symbol and a small unit of data. For example, each symbol may encode one or several binary digits or 'bits'. The data may also be represented by the transitions between symbols, or even by a sequence of many symbols.

The **symbol duration time**, also known as unit interval, can be directly measured as the time between transitions by looking into an eye diagram of an oscilloscope. The symbol duration time T_s can be calculated as:

$$T_s = 1/f_s$$

where f_s is the symbol rate.

> **A simple example**: A baud rate of 1 kBd = 1,000 Bd is synonymous with a symbol rate of 1,000 symbols per second. In the case of a modem, this corresponds to 1,000 tones per second, and in the case of a line code, this corresponds to 1,000 pulses per second. The symbol duration time is 1/1,000 second = 1 millisecond.

Delay Spread

In telecommunications, the delay spread is a measure of the multipath richness of a communications channel. In general, it can be interpreted as the difference between the time of arrival of the earliest significant multipath component (typically the line-of-sight component) and the time of arrival of the last multipath components.

Denoting the power delay profile of the channel by $A_c(\tau)$ the mean delay of the channel is

$$\tau = \frac{\int_0^\infty \tau A_c(\tau) d\tau}{\int_0^\infty A_c(\tau) d\tau}$$

Resources

https://en.wikipedia.org/wiki/Symbol_rate

https://en.wikipedia.org/wiki/Delay_spread

Experiment 1 Instructions – Inter-Symbol Interference

In the following, we assume the reader is familiar with OFDM and how it works.

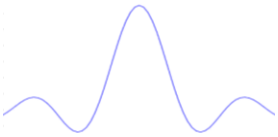
1. Open a new QT GUI .grc file and add a float Vector Source.
   Under Vector input a vector of a length of 64 zero bits with the 32nd bit set to one (000...0001000..000):

   31*[0,]+[1,]+32*[0,]

   This vector represents a single symbol and we would like it to constantly repeat itself.
   Set Repeat to Yes.

2. The symbol we would like to use is raised cosine (which represents a real symbol that passes in a system), in order to do that, place an Interpolation FIR Filter with 4 **Interpolation**s (interpolation is a type of estimation, a method of constructing new data points within the range of a discrete set of known data points) and a **Tap** of

   firdes.root_raised_cosine(4,4,1,0.25,4*45)

3. We would like to create a sequence of symbols so we split the base symbol into several symbols each with a different time of transmission.
   To create this, you can add several Delay modules (5 for example) and connect the symbol to each Delay module input.

4. We would like to be able to control the distance between each symbol, so we can create a QT GUI Range called symbol_guard ranging from 0 to 200 with a step of 1 and a default value of 5.

5. In each symbol's delay enter the symbol_guard*symbol number as the delay (for example the first symbol will have a delay of symbol_guard*1, the second symbol will have a delay of symbol_guard*2, and so forth). As we increase the symbol_guard the gap between each symbol will increase.

6. By definition, the symbol created represents a "logical one". You can add a Multiply Const with a constant of -1 module for each symbol you want to be "logical zero". For each symbol you want to remain the same you can bypass the multiply const block.

7. Connect all symbols signals to a QT GUI Time Sync with a number of ports as per the number of symbols you created and leave another port for the sum of the signals.

8. Connect all signals to an Add module and connect its output to the last port of the QT GUI time sink. Connect the output to a Virtual Sink as well and name the stream ID as "Symbol Source", this will be the sequence we will transmit.

9. We now want to use our Symbol Source over a medium with a controllable delay spread. Place a Virtual Source corresponding to the Virtual Sink with the stream ID as "Symbol Source".

10. Create a QT GUI Range called delay_spread [0,50] with steps of 1 and with a default value of 1.

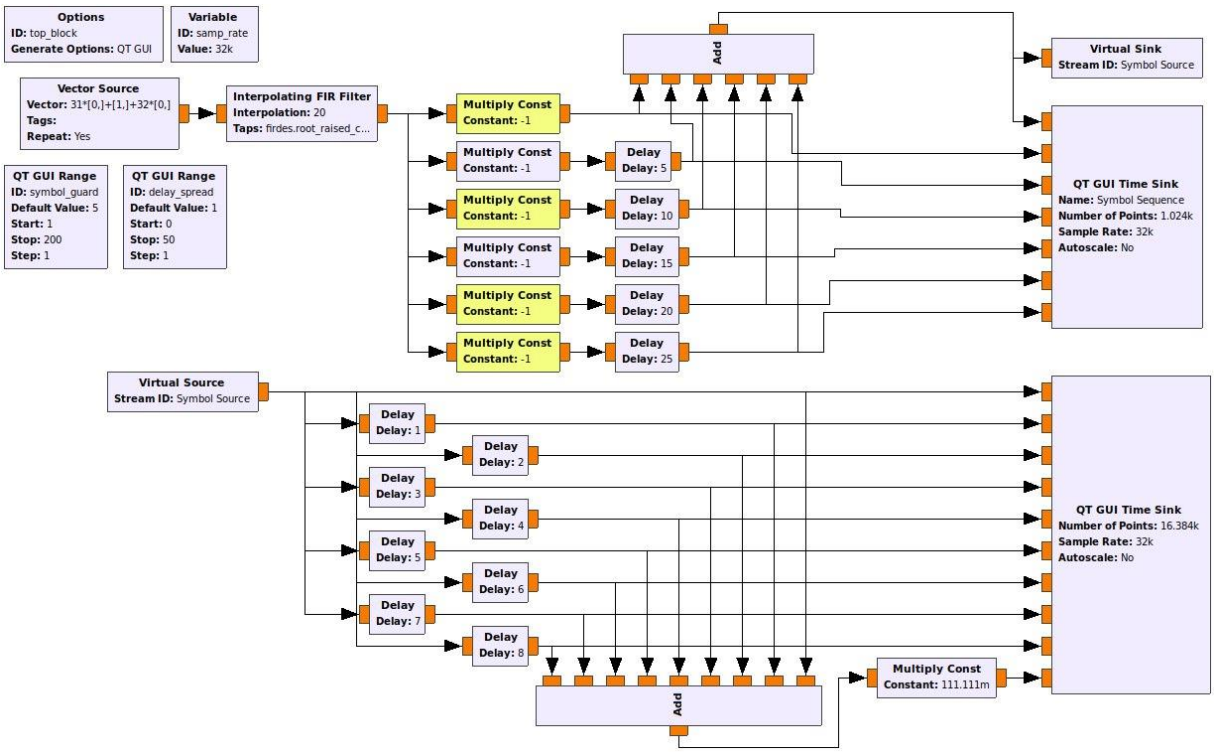
11. To the Virtual Sink connect 8 delay modules and in each enter the Delay parameter delay_spread, delay_spread*2, delay_spread*3, …, delay_spread*8 accordingly. This will represent the time each ray is delayed. The slowest ray will take delay_spread*8 and the fastest ray will have no delay.

12. Create a QT GUI Time SInk with 10 ports and connect all delayed rays as well as a ray that has no delay (9 rays in total), The 10th port will be connected to the sum of all rays.

13. Connect all rays to a 9 port “Add” module and connect its output to the QT GUI Time Sink Via a Multiply Const module (Multiply by 0.111).

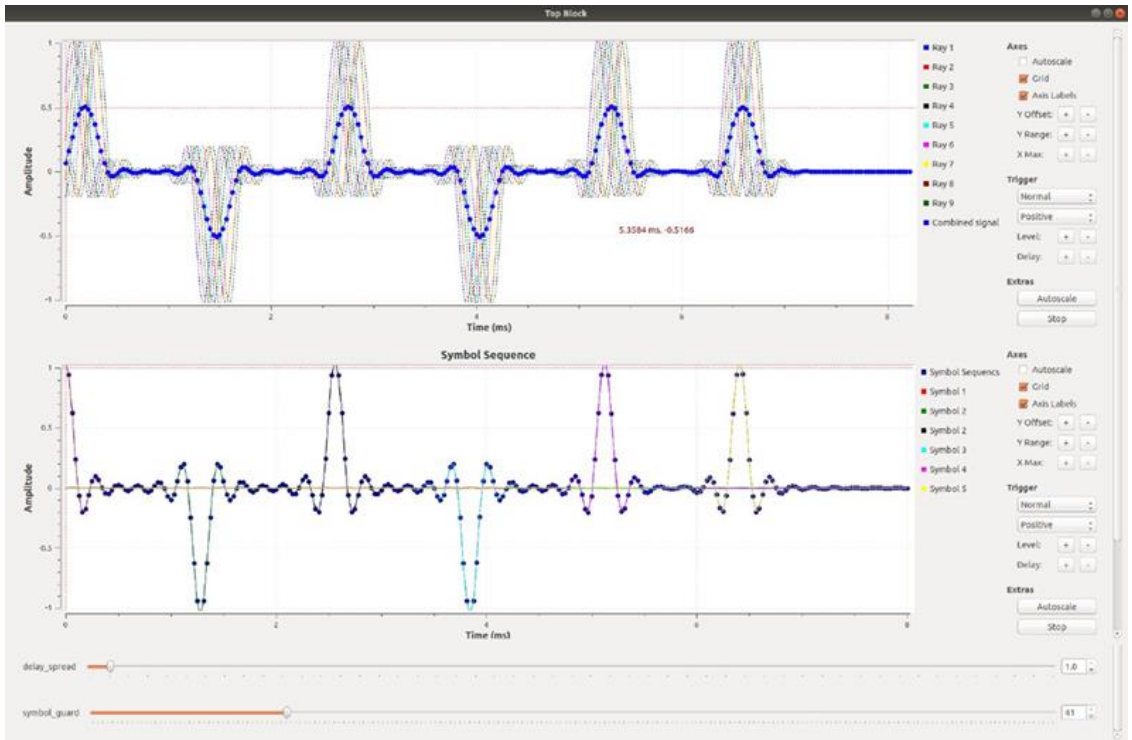
14. Set your GUI hints as you wish to show them properly. For your convenience add a Control Panel to all QT GUIs, and set the trigger to Normal with a Level of 0.5. This will display the symbol sequence properly.

If all steps are correctly done your code should look similar to this:



*Notice, the blocks in yellow are bypassed.

Your GUI should look similar to this:



Experiment 1 Questions

1. Change the delay spread slider and the symbol guard slider to achieve the following:
   a. A scenario where there is noticeable inter-symbol interference.
   b. A scenario where there is **no** noticeable inter-symbol interference.

   Show two different cases for both scenarios (2 of a & 2 of b).

   Explain each scenario outcome (why it occurs and how it affects the reception of data).

2. Find an expression for the needed symbol guard according to the signal's delay spread, T_m. Assume the symbols are considered as interfering with each other when 10% of the symbols overlap in time, and the symbol duration is denoted as T_s and the symbol rate as R. **Explain** your results. **Hint:** the total time a symbol is in the air is $T_s + T_m$.

3. In our experiment, we control the delay spread. Describe and **explain** the connection between the delay spread and the coherence bandwidth. Calculate the possible coherence bandwidth range possible due to the range of the delay spread in our experiment.

4. The time between symbols is dedicated to assisting in differentiating consecutive symbols in a large delay spread scenario, in our experiment this time was left empty (without data). Suggest, describe, and **explain** a method in which the time between symbols could be better used to assist the differentiation of symbols.

5. What are the possible fading types possible due to the multipath delay spread? Are there other types of Fading possible? If so, what are they, and give a numerical example to each? **Explain**.

Part 2 – Fading

In wireless communications, fading is a variation of the attenuation of a signal with various variables. These variables include time, geographical position, and radiofrequency. Fading is often modeled as a random process. A fading channel is a communication channel that experiences fading. In wireless systems, fading may either be due to multipath propagation, referred to as multipath-induced fading, weather (particularly rain), or shadowing from obstacles affecting the wave propagation, sometimes referred to as shadow fading.

The presence of reflectors in the environment surrounding a transmitter and receiver creates multiple paths that a transmitted signal can traverse. As a result, the receiver sees the superposition of multiple copies of the transmitted signal, each traversing a different path. Each signal copy will experience differences in attenuation, delay, and phase shift while traveling from the source to the receiver. This can result in either constructive or destructive interference, amplifying or attenuating the signal power seen at the receiver. Strong destructive interference is frequently referred to as a deep fade and may result in temporary failure of communication due to a severe drop in the channel signal-to-noise ratio.

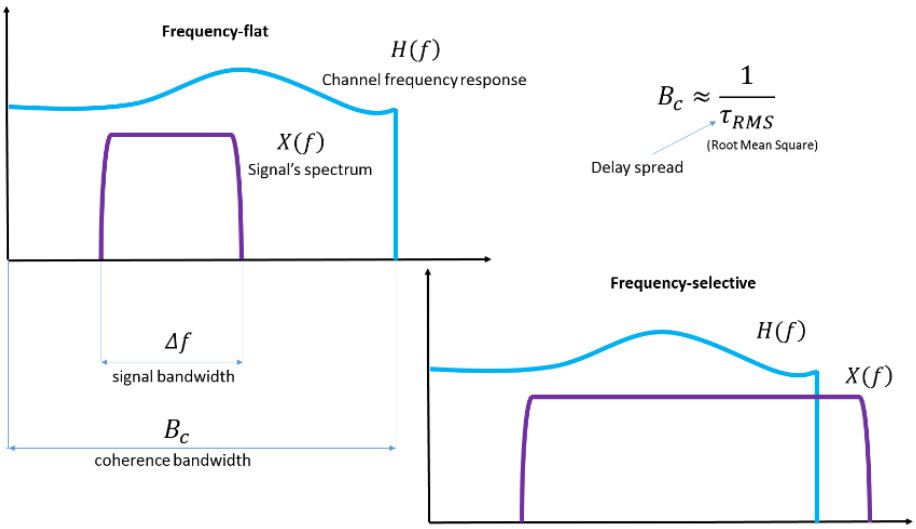
Fading channel models are often used to model the effects of electromagnetic transmission of information over the air in cellular networks and broadcast communication.

Slow fading arises when the coherence time of the channel is large relative to the delay requirement of the application. In this regime, the amplitude and phase change imposed by the channel can be considered roughly constant over the period of use. Slow fading can be caused by events such as shadowing, where a large obstruction such as a hill or large building obscures the main signal path between the transmitter and the receiver. The received power change caused by shadowing is often modeled using a log-normal distribution with a standard deviation according to the log-distance path loss model.

Fast fading occurs when the coherence time of the channel is small relative to the delay requirement of the application. In this case, the amplitude and phase change imposed by the channel varies considerably over the period of use.

Selective fading or **frequency selective fading** is a radio propagation anomaly caused by partial cancellation of a radio signal by itself — the signal arrives at the receiver by two different paths, and at least one of the paths is changing (lengthening or shortening). This typically happens in the early evening or early morning as the various layers in the ionosphere move, separate, and combine. The two paths can both be skywave or one be groundwave.

Selective fading manifests as a slow, cyclic disturbance; the cancellation effect, or "null", is deepest at one particular frequency, which changes constantly, sweeping through the received audio.

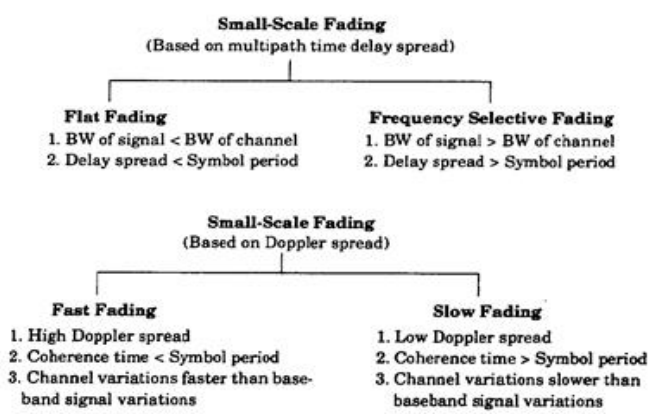


As the carrier frequency of a signal is varied, the magnitude of the change in amplitude will vary. The coherence bandwidth measures the separation in frequency after which two signals will experience uncorrelated fading.

In flat fading, the coherence bandwidth of the channel is larger than the bandwidth of the signal. Therefore, all frequency components of the signal will experience the same magnitude of fading.

In frequency-selective fading, the coherence bandwidth of the channel is smaller than the bandwidth of the signal. Different frequency components of the signal, therefore, experience e uncorrelated fading.

The following figure summarizes the four different types of fading:



Resources

Wireless communications, Andrea Goldsmith, Stanford University, California, 2005, 9780511841224

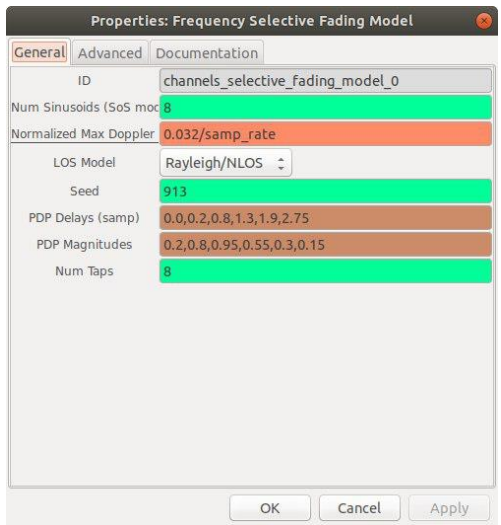
https://www.site.uottawa.ca/~sloyka/elg4179/Lec_5_ELG4179.pdf

Rappaport Wireless Communications Principles And Practice 2Nd Edition, Prentice Hall, 2002
Theodore S. Rappaport, 978-0130422323

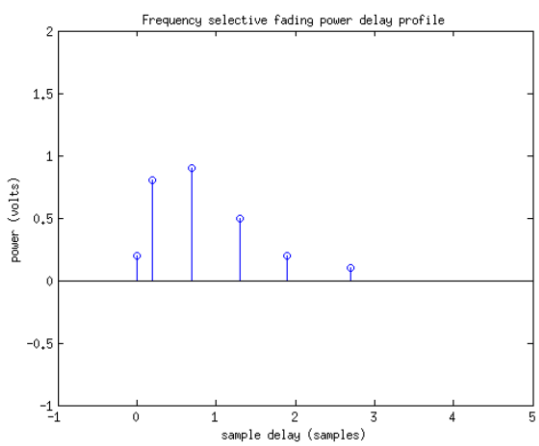
Experiment 2 – Realistic Fading Simulation

In GNU Radio we can define a fixed Power Delay Profile (PDP) in terms of fractional sample times and powers using one of the channel models that GNU offers. Let's also combine Frequency-selective time-varying fading for our signal.

- Add a Signal Source of 1KHz Cosine with an amplitude of 1V.
- Add a Fast Noise Source with an amplitude of 100mV.
- Combine both using "Add" and connect to a "Frequency selective channel model" component with the following preferences:

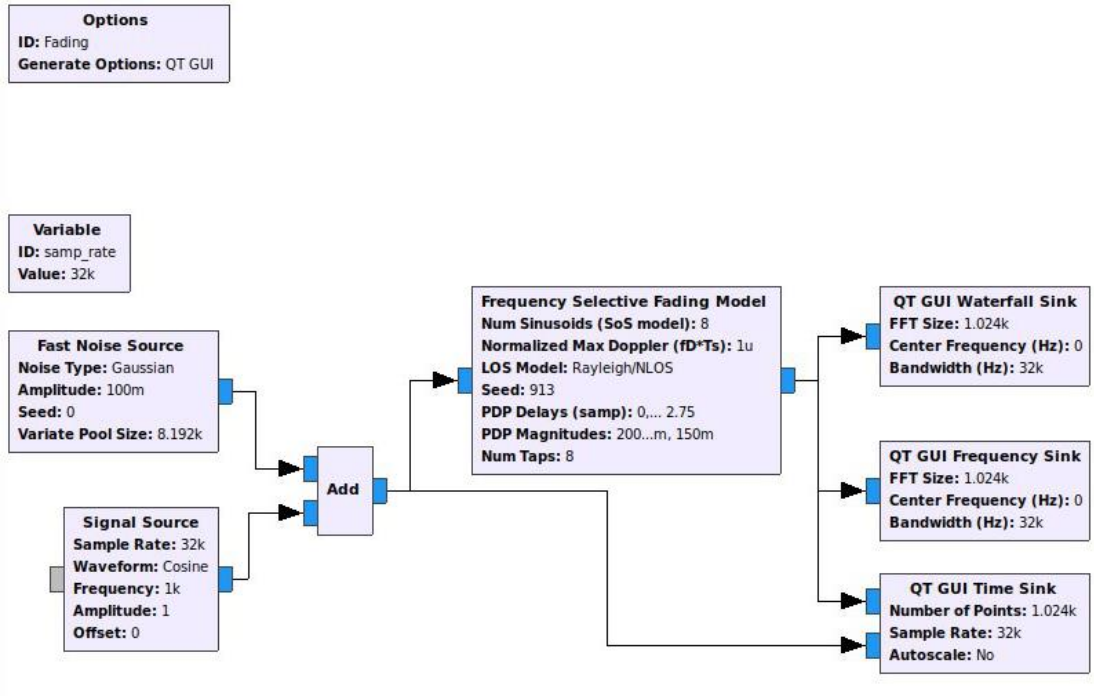


The Power Delay Profile values represent a more realistic model so we can see more fading multipath effects:

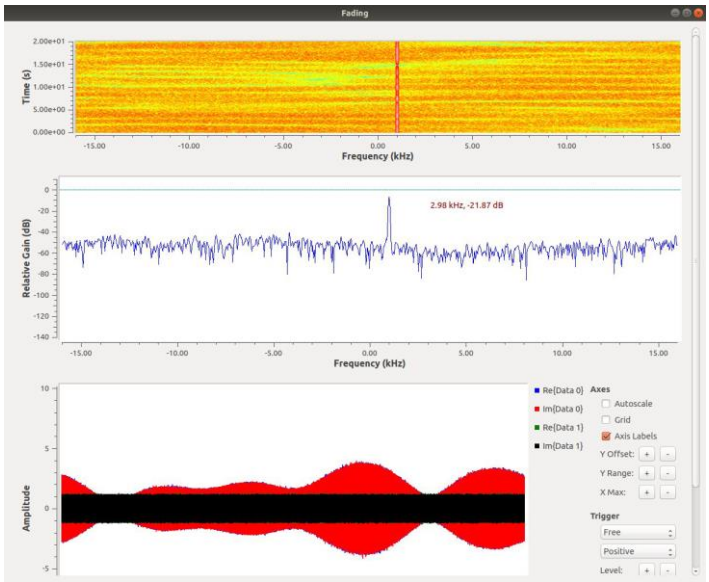


- Place a 2-input QT GUI Time Sink (with config panel) and connect to the signal before and after the channel model.
- Place a QT GUI Frequency Sink and connect to the signal after the channel model.
- Place a QT GUI Waterfall Sink and connect to the signal after the channel model.

If you performed all the stages correctly, your system's code should look like this:



These are the signals that you should see in your GUI:



At this stage, you can notice the phenomena of frequency selective fading on our signal. Since the signal is a narrow continuous cosine wave it is affected only by the changes in amplitude where he occupies the BW.

But what if our signal's BW was wide? Let's change our signal to something wider...

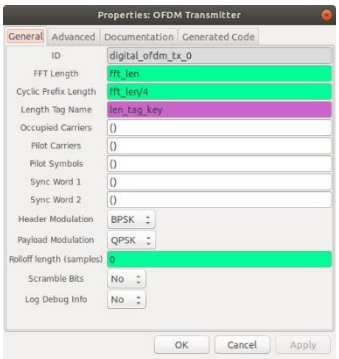
Choose a modulation scheme as you wish (e.g., BPSK from GNURadio Wiki, ASK from lab 1, <u>not OFDM</u>) and try to transfer data through this Frequency Selective Fading Channel model.
Attach your .grc code and explain what occurred (you might find it difficult under this channel model so first try to Bypass it and only when the data is flowing, add the Channel model again). You should show us distorted signals/symbols

Well, since under the frequency selective fading channel model we could (possibly) not recover the data properly, let us try something that was designed just for such harsh conditions - OFDM!

- Change the signal source to a File source and point to the Alice.txt text file.
- Update the samp_rate to 320Ksps.
- Create the following variables:
  - fft_len - 64
  - packet_len – 50
  - len_tag_key – packet_len
- Add a "Stream To Tagged Stream" module and set packet_len as the packet length and len_tag_key as the Length of the Tag Key.
- Add an OFDM Transmitter (Modulator) and feed the File source stream to it. The modulator's output should then be connected to the Add module.

  Set the OFDM Transmitter parameters as follow:

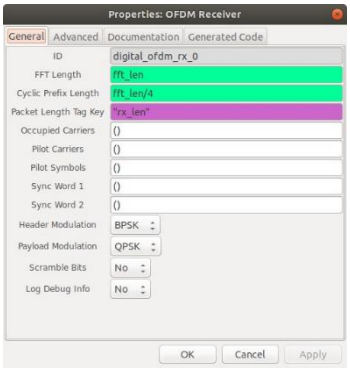
  - Payload Modulation: QPSK
  - Header Modulation: BPSK
  - FFT length :fft_len
  - Cyclic Prefix: fft_len/4



- After the noise addition and frequency selective fading channel model add a Throttle with Sample rate same as samp_rate and connect its output to an OFDM Receiver (Demodulator).

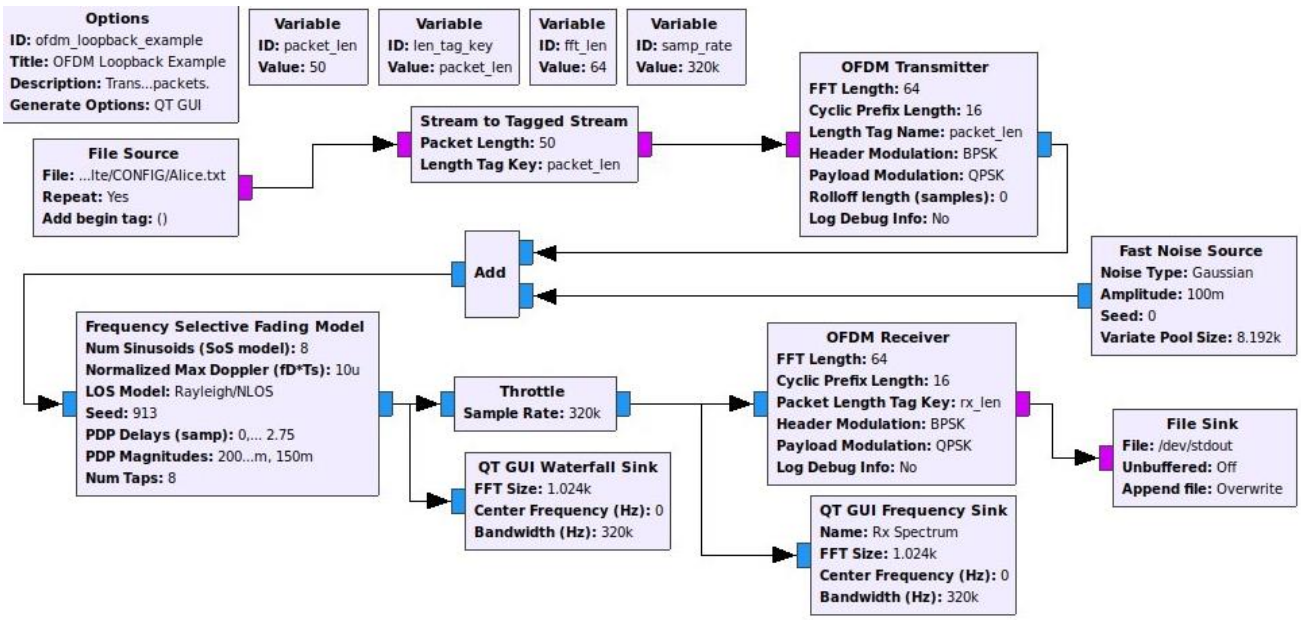
  Set the OFDM Receiver parameters as follows:

  - Payload Modulation: QPSK
  - Header Modulation: BPSK
  - FFT length :fft_len
  - Cyclic Prefix: fft_len/4

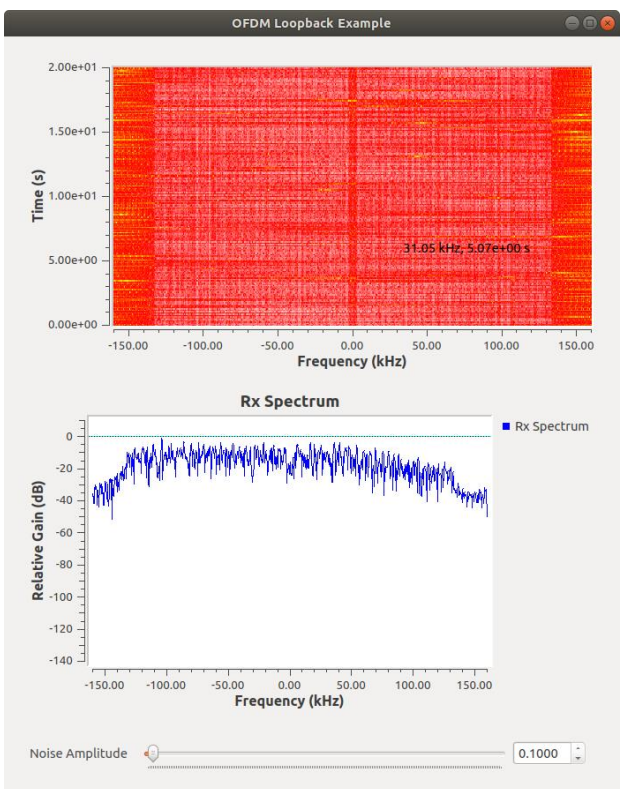


Connect the receiver's output to a File Sink with a File /dev/stdout.

If you performed all the stages correctly, your system's code should look like this:



These are the signals that you should see in your GUI:



Notice that the signal is “breathing” (changes amplitude in different frequencies in time).

Even in these harsh channel conditions, we receive data, Alice.txt should be nearly flawlessly printed during this transmission. But why is that?

Save your code and add it to your submission!

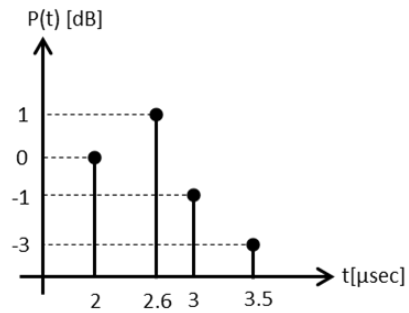
Experiment 2 Questions –

1. **Explain** what coherence time and coherence bandwidth are.

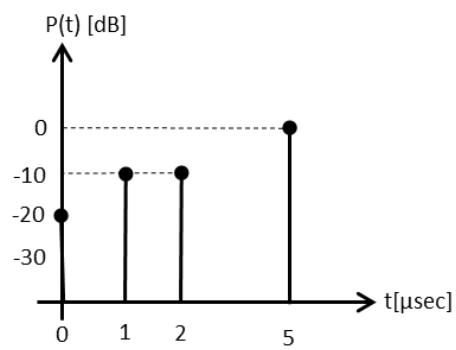
2. **Explain** how multi-path propagation can cause frequency-selective fading and how multi-path propagation can cause a different kind of fading.

3. In a typical urban area, the multipath delay spread of a wireless channel is $T_d \approx 5\mu sec$. The bandwidth BW of the transmitted signal is $BW = 1MHz$. Is it a frequency-selective channel? Determine its coherence bandwidth. **Explain** your results.

4. Will a 50kHz signal experience flat or selective fading in the following channel?



5. Calculate the mean excess delay, and RMS delay spread for the multipath profile given in the figure below. Estimate the coherence bandwidth of the channel.



Part 3 – Orthogonal Frequency-Division Multiplexing

Background

Orthogonal frequency division multiplexing (OFDM) is a special case of multicarrier transmission, where a single data stream is transmitted over a number of lower-rate subcarriers. In July 1998, the IEEE standardization group decided to select OFDM as the basis for their new 5-GHz IEEE 802.11a standard (Wi-Fi), targeting a range of data streams from 6 up to 54 Mbps. This new standard is the first one to use OFDM in packet-based communications, while the use of OFDM until now was limited to continuous transmission systems.

IEEE 802.11 (Wi-Fi) stations communicate by sending each other data packets: blocks of data individually sent and delivered over the radio. As with all radio, this is done by the modulating and demodulating of carrier waves. Wi-Fi 802.11a, Wi-Fi 4, 5, and 6 use multiple carriers on frequencies within the channel (OFDM).

OFDM Basics

OFDM is a multi-carrier modulation scheme that is widely adopted in many recently standardized broadband communication systems due to its ability to cope with **frequency selective fading**. The main idea behind OFDM is to divide a high-rate encoded data stream (with symbol time T_S) into N parallel sub-streams (with symbol time $T = N \cdot T_S$) that are modulated onto N orthogonal carriers (referred to as subcarriers).

If N subcarriers are used, and each subcarrier is modulated using M alternative symbols, the OFDM symbol alphabet consists of M^N combined symbols. The low-pass equivalent OFDM signal is expressed as

$$v(t) = \sum_{k=0}^{N-1} X_k e^{\frac{j2\pi kt}{T}} \quad , \quad 0 \le t \le T$$

Where $\{X_k\}$ are the data symbols, N is the number of subcarriers, and T is the OFDM symbol time. The subcarrier spacing of $\frac{1}{T}$ makes them orthogonal over each symbol period; this property is expressed as:

$$\frac{1}{T}\int_0^T \left(e^{\frac{j2\pi k_1 t}{T}}\right)^* \left(e^{\frac{j2\pi k_2 t}{T}}\right) dt = \frac{1}{T}\int_0^T e^{\frac{j2\pi(k_2-k_1)t}{T}}\, dt = \delta_{k_1 k_2}$$

where $(\cdot)^*$ denotes the complex conjugate operator and δ, is the Kronecker delta.

This operation is easily implemented in the discrete-time domain through an N-point inverse discrete Fourier transform (IDFT) unit and the result is transmitted serially. At the receiver, the information is recovered by performing a DFT on the received block of signal samples.

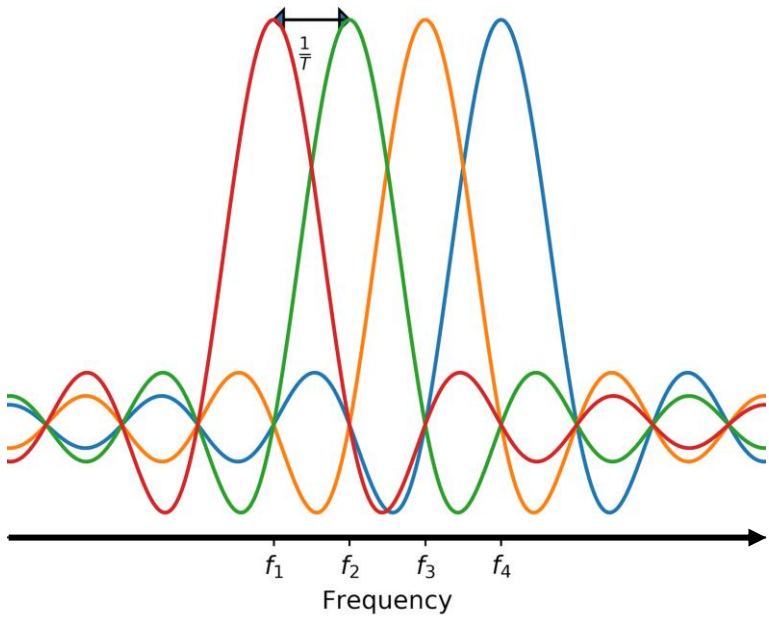


Figure 1- Subcarriers system of OFDM signals after FFT

An OFDM carrier signal is the sum of a number of orthogonal subcarriers, with baseband data on each subcarrier being independently modulated commonly using some type of quadrature amplitude modulation (QAM) or phase-shift keying (PSK). This composite baseband signal is typically used to modulate the main RF carrier.

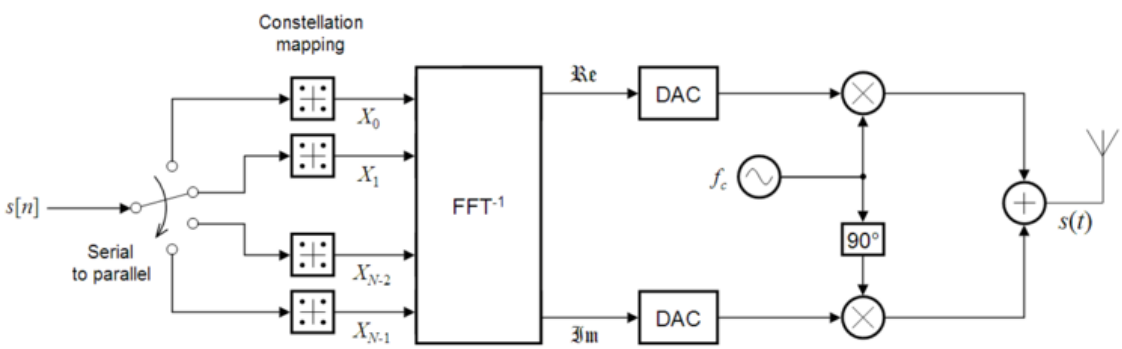


Figure 2 - OFDM Transmitter

$s[n]$ is a serial stream of binary digits. By inverse multiplexing, these are first demultiplexed N parallel streams, and each one mapped to a (possibly complex) symbol stream using some modulation constellation (QAM, PSK, etc.). Note that the constellations may be different, so some streams may carry a higher bit rate than others.

An inverse FFT is computed on each set of symbols, giving a set of complex time-domain samples. These samples are then quadrature-mixed to passband in the standard way. The real and imaginary components are first converted to the analog domain using digital-to-analog converters (DACs); the analog signals are then used to modulate cosine and sine waves at the carrier frequency, f_c, respectively. These signals are then summed to give the transmission signal, $s(t)$.

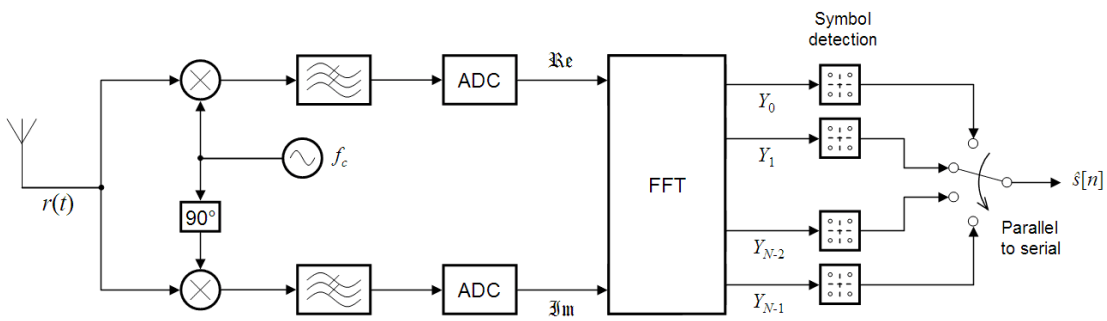


Figure 3 - OFDM Receiver

The receiver picks up the signal $r(t)$, which is then quadrature-mixed down to baseband using cosine and sine waves at the carrier frequency. This also creates signals centered on $2f_c$, so low-pass filters are used to reject these. The baseband signals are then sampled and digitized using analog-to-digital converters (ADCs), and a forward FFT is used to convert back to the frequency domain.

This returns N parallel streams, each of which is converted to a binary stream using an appropriate symbol detector. These streams are then recombined into a serial stream, $s[n]$, which is an estimate of the original binary stream at the transmitter.

Resources

Standard IEEE 802.11a – Orthogonal Frequency Division Multiplexing for Wireless Networks
https://en.wikipedia.org/wiki/Orthogonal_frequency-division_multiplexing

Wireless communications, Andrea Goldsmith, Stanford University, California, 2005, 9780511841224

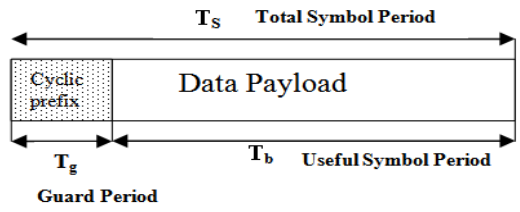
Theoretical Questions

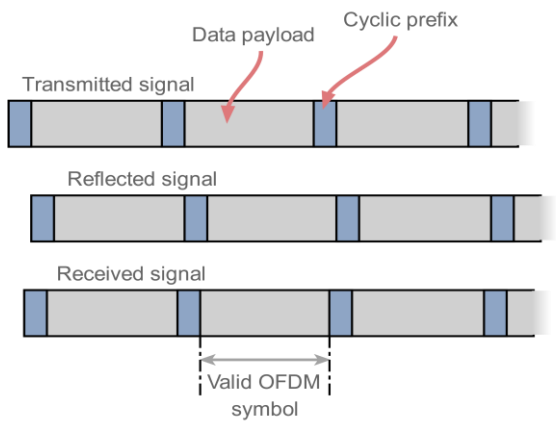
1. What is the motivation for using OFDM as a modulation technique for time-dispersive channels compared to a single-carrier system? Explain.
2. How does dividing a data stream signal into N parallel sub-streams help coping with **frequency selective fading** compared to a single wide stream? Explain.
3. What is the desired relation($<, >, =, \leq, \geq, \ll, \gg$) between:
    a. length of multicarrier symbol and coherence time
    b. length of multicarrier symbol and multipath delay spread
4. Give a brief explanation of DFT/IDFT-based multicarrier modulation answering the following questions:
    a. What are the basis functions?
    b. How does the IDFT modify the basis functions so that they carry the information?
    c. How does the DFT extract the information from the received signal?

Cyclic Prefix

To avoid inter-symbol interference in multipath fading channels, a guard interval of length T_g is inserted prior to the OFDM block. During this interval, a cyclic prefix is transmitted such that the signal in the interval $-T_g \leq t \leq 0$ equals the signal in the $(T - T_g) \leq t < T$.
The OFDM signal with a cyclic prefix is thus:

$$v(t) = \sum_{k=0}^{N-1} X_k e^{\frac{j2\pi kt}{T}} \quad , -T_g \leq t \leq T$$





The cyclic prefix performs two main functions.

- The cyclic prefix provides a guard interval to eliminate inter-symbol interference from the previous symbol.
- It repeats the end of the symbol so the linear convolution of a frequency-selective multipath channel can be modeled as circular convolution, which in turn may transform to the frequency domain via a discrete Fourier transform. This approach accommodates simple frequency-domain processing, such as channel estimation and equalization.

The base-band signal $v(t) = \sum_{k=0}^{N-1} X_k e^{\frac{j2\pi kt}{T}} \quad , -T_g \leq t \leq T$ can be either real or complex-valued.

For wireless applications, the base-band signal is typically complex-valued; in which case, the transmitted signal is up-converted to a carrier frequency f_c. In general, the transmitted signal can be represented as:

$$s(t) = R\{v(t)e^{j2\pi f_c t}\} = \sum_{k=0}^{N-1} |X_k| \cos\left(2\pi\left[f_c + \frac{k}{T}\right]t + arg\ [X_k]\right)$$

This signal is transmitted wirelessly.

Resources
A tutorial on the significance of Cyclic Prefix (CP) in OFDM Systems.

Theoretical Questions

5. What are the advantages and what are the drawbacks of using a cyclic prefix?
6. Why is it beneficial to use the DFT/IDFT transform-pair in combination with a long-enough cyclic extension (cyclic-prefix extension after the IDFT and cyclic-prefix removal before DFT) compared to other transforms?
7. Consider a 20 MHz OFDM system. The length of one OFDM symbol including the cyclic prefix is 4 µs. 20% of this is made up of the cyclic prefix itself.

    A. What subcarrier spacing is used? How many subcarriers does the system use?

    B. A packet consists of 2 OFDM symbols used as a preamble (not used for data transmission) and 100 OFDM symbols of data. However, 4 subcarriers are NOT used for data transmission as they are needed as pilots. Given that all other subcarriers are used for data transmission, calculate the number of bits that can be transmitted in one packet, if 4-QAM or 64-QAM is used.

    C. What is the maximum channel delay that can be corrected by the cyclic prefix?

    D. Due to the strong sidelobes of OFDM, we now demand a guard band, a frequency range of 1.25 MHz at both ends of the spectrum to be kept empty. How many subcarriers do we lose. Calculate the throughput that we can now achieve (preamble and pilots stay the same).

    E. To combat outdoor channels, bandwidth is reduced to 10MHz. The number of subcarriers and percentage of cyclic prefix stay exactly as they are. How long is the cyclic prefix now? How much longer can the longest multipath be than the line of sight?

Part 4 – OFDM Tx\Rx system

Description

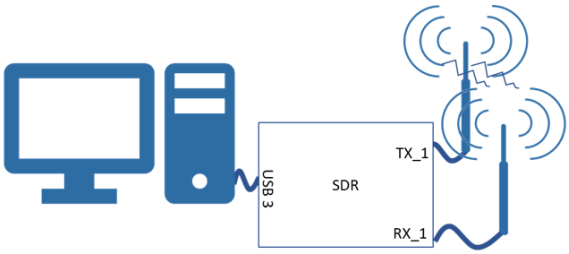
In this experiment, we will use an OFDM communication system using the GNU Radio freeware that will transmit a text file over the air. We will then change the channel model parameters to observe the upper bounds of the OFDM system's capabilities.

Equipment needed

- 1 Linux PC with GNU Radio installed.
- 1 LimeSDR/USRP Software-defined radios.
- 2 SMA Antennas

If any of the above equipment is missing or defective, notify the lab instructor prior to starting this session, or otherwise, you may obtain false results (and major frustration).

Equipment Setup



Recording

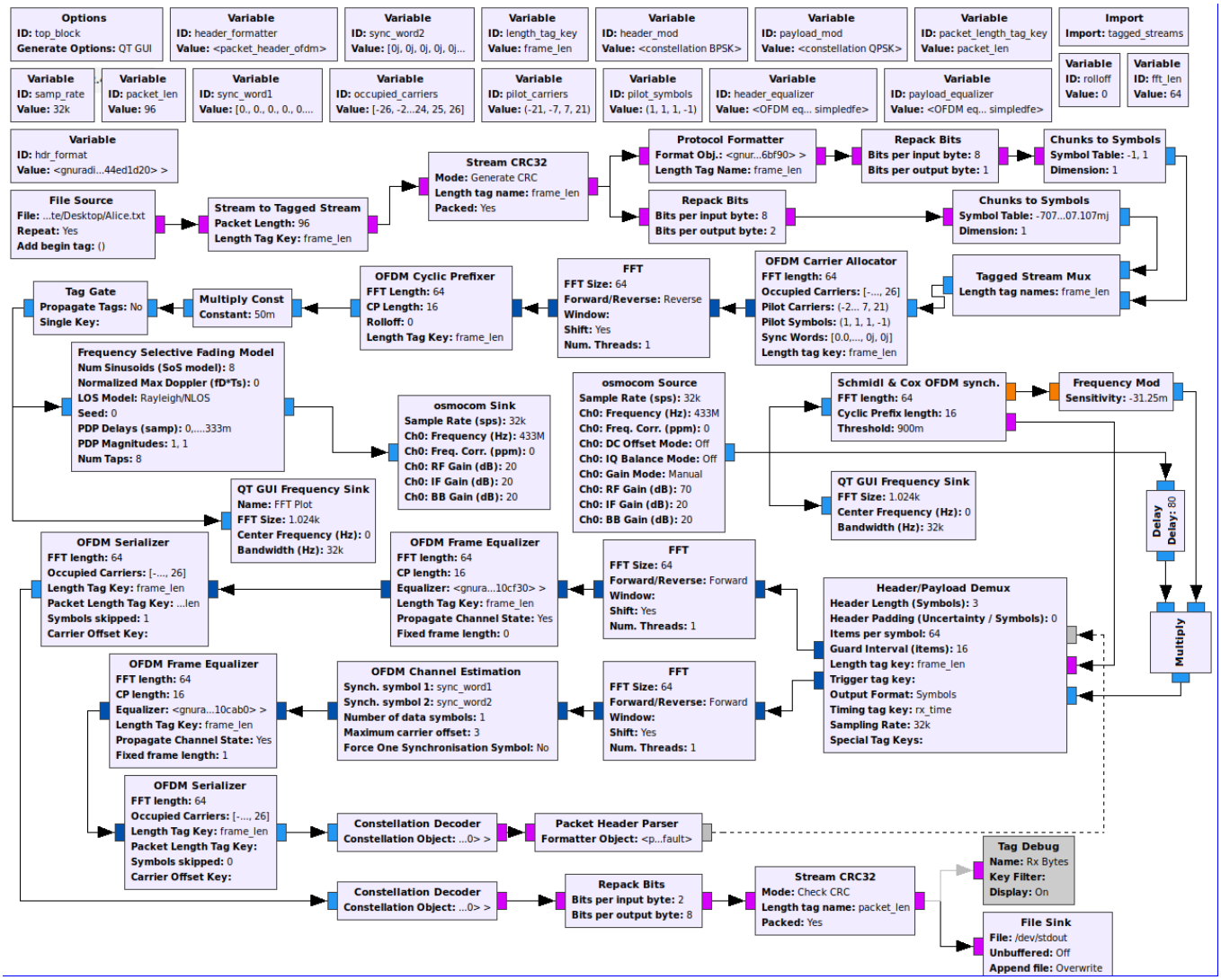
This is a practical session; thus, you most likely do not have this equipment at your disposal at home other than currently. Therefore, it is highly recommended that you document and record your results during this session. This will assist you in completing your report at home. You may not fully complete what is required of you during the time you have, take this into account.

The OFDM system

Open the provided OFDM system.

Change the transmission frequency to $433MHz + (\#PC\%4) * 2MHz$

Make sure that all the components are connected, and your code looks like this:



OFDM Transmitter

The Data

Our data is a text file of *Alice in Wonderland* in a format of a Byte (8 bits per character). The data is streamed at the sample rate of 32Ksps. This stream of data is then Tagged with a “frame_len” (value of 96 characters), this is done in order to assist in the recognition of packet start and end during the OFDM modulation.

To each data packet, we add a CRC32 in order to check for packet corruption at the receiver side.

The stream is then split into a header stream and a payload stream. The Protocol Formatter module uses a header format object to create a header from a tagged stream packet.

The payload is sent as a parallel tagged stream and is later muxed together with the header stream.

Prior to muxing the streams, the header data stream is repacked as BPSK symbols {-1,1} and the data payload stream is repacked as QPSK symbols $\{\pm\sqrt{2} \pm \sqrt{2} \cdot j\}$.

Carrier and Symbol Allocation

Many blocks require knowledge of which carriers are allocated, and whether they carry data or pilot symbols. GNU Radio blocks use three objects for this, typically called occupied_carriers (for the data symbols), pilot_carriers, and pilot_symbols (for the pilot symbols).

Every one of these objects is a vector of vectors. occupied_carriers and pilot_carriers identify the position within a frame where data and pilot symbols are stored, respectively.

occupied_carriers[0] identifies which carriers are occupied on the first OFDM symbol, occupied_carriers[1] do the same on the second OFDM symbol, etc.

Here's an example:

```
occupied_carriers = ((-2, -1, 1, 3), (-3, -1, 1, 2))
```

```
pilot_carriers = ((-3, 2), (-2, 3))
```

Every OFDM symbol carries 4 data symbols. On the first OFDM symbol, they are on carriers -2, -1, 1, and 3. Carriers -3 and 2 are not used, so they are where the pilot symbols can be placed. On the second OFDM symbol, the occupied carriers are -3, -1, 1, and 2. The pilot symbols must thus be placed elsewhere, and are put on carriers -2 and 3.

If there are more symbols in the OFDM frame than the length of occupied_carriers or pilot_carriers, they cyclically wrap-around (in this example, the third OFDM symbol uses the allocation in occupied_carriers[0]).

But how are the pilot symbols set? This is a valid parametrization:

```
pilot_symbols = ((-1, 1j), (1, -1j), (-1, 1j), (-1j, 1))
```

The position of these symbols is those in pilot_carriers. So on the first OFDM symbol, carrier -3 will transmit a -1, and carrier 2 will transmit a 1j. Note that pilot_symbols is longer than pilot_carriers in the example - this is valid, the symbols in pilot_symbols[2] will be mapped according to pilot_carriers[0].

Carrier Indexing

Carriers are always indexed starting at the DC carrier, which has the index 0 (you usually don't want to occupy this carrier). The carriers right off the DC carrier (the ones at higher frequencies) are indexed with 1 through $\frac{N}{2} - 1$ (N being the FFT length again).

The carriers left of the DC carrier (with lower frequencies) can be indexed $-\frac{N}{2}$ through -1 or $\frac{N}{2}$ through $N - 1$. Carrier indices $N - 1$ and -1 are thus equivalent. The advantage of using negative carrier indices is that the FFT length can be changed without changing the carrier indexing.

IFFT with FFT Shifting

After all, symbols were allocated to a corresponding carrier we apply an Inverse FFT of the size 64 samples with a shift.

In all cases where OFDM symbols are passed between blocks, the default behavior is to FFT-Shift these symbols, i.e. that the DC carrier is in the middle (to be precise, it is on the carrier floor($\frac{N}{2}$) where N is the FFT length and carrier indexing starts at 0).

The reason for this convention is that some blocks require FFT-shifted ordering of the symbols to function (such as gr::digital::ofdm_chanest_vcvc), and for consistency's sake, this was chosen as a default for all blocks that pass OFDM symbols. Also, when viewing OFDM symbols, FFT-shifted symbols are in their natural order, i.e. as they appear in the passband.

Cyclic prefix

The OFDM Cyclic Prefixer adds a cyclic prefix of a length of 16 samples and performs pulse shaping on the OFDM symbols. The pulse shape is a raised cosine in the time domain.

Transmission

After multiplying by a constant of 0.05 and stopping the propagation of the tag the OFDM stream is applied with a **frequency selective channel model** and transmitted at the selected frequency of 433MHz with a gain of 20 using the SDR.

OFDM Receiver

Synchronization

Given the received OFDM signal, how can one know at which point in time the OFDM symbols are located? Or, equivalently, on which signal part the receiver needs to perform the FFT?

A common methodology used for OFDM has been published by Schmidl & Cox in 1997. Their proposal is to estimate the start of each OFDM symbol.

The Schmidl & Cox metric is based on an OFDM symbol $x[n]$ that contains two repetitive parts of length $\frac{k}{2}$, i.e. $x[n] = x[n + \frac{k}{2}]$. To achieve a repetition, we leverage a DFT property: Upsampling in frequency domain corresponds to repetition in time domain. This results in a repetition in time that allows to synchronize the OFDM symbols.

Upon reception of the OFDM, radio signal is sent to a Schmidl and Cox synchronization mechanism which generates a detect signal that indicates the start of the OFDM frame according to the FFT length and the Cyclic prefix length. The mechanism also generates coarse frequency correction and channel estimation to synchronize the reception of the original signal delayed by 80 samples (the time needed for the Schmidl and Cox module to synchronize the frame)

Header and Payload Demuxing

Once the signal is synchronized the header symbols and the payload symbols are separated according to the OFDM known timing parameters (such as FFT length and Cyclic prefix length). The output is two streams of data - a data stream containing the frame header data and a data stream containing the payload data.

FFT shift

An FFT of the size 64 samples with a shift is applied to each data stream. The FFT shifts the OFDM symbols into the frequency domain, where the signal processing is performed (the OFDM frame is thus in the memory in matrix form).

Channel Estimation

For the header stream, the FTT shifted stream is passed to a block that uses the preambles to perform channel estimation and coarse frequency offset. Both of these values are added to the output stream as tags; the preambles are then removed from the stream and not propagated.

Frame Equalizer

Both the coarse frequency offset correction and the equalizing (using the initial channel state estimate) is done in the following block, gr::digital::ofdm_frame_equalizer_vcvc. The interesting property about this block is that it uses a gr::digital::ofdm_equalizer_base derived object to perform the actual equalization.

Serializer

The last block in the frequency domain is the gr::digital::ofdm_serializer_vcc, which is the inverse block to the carrier allocator. It plucks the data symbols from the occupied_carriers and outputs them as a

stream of complex scalars. These can then be directly converted to bits or passed to a forward error correction decoder.

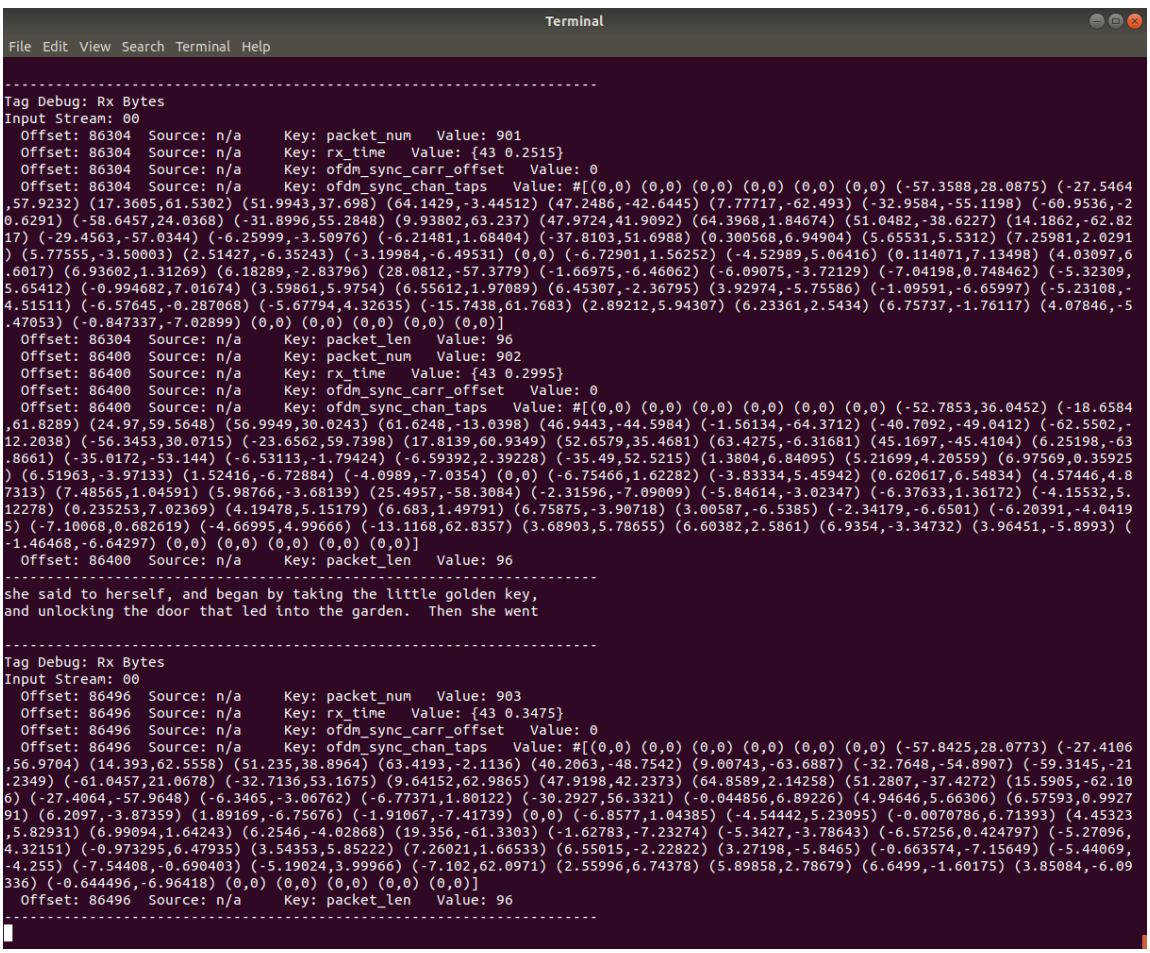
Constellation Decoder and Data output

For each stream the symbols are decoding accordingly (BPSK for the header and OPSK for the payload) and translated to their corresponding bits which are then repacked as chars (for the payload), CRC checked for error, and printed to the command prompt.

The header recovered data is parsed and feedback to the Header/Payload Demux to allow the demuxing according to two synchronization words embedded in the date during the OFDM modulation at the transmitter side.

Debug

The Tag Debug module allows the printing of the tags attached to the OFDM stream. This shows the carriers/symbols used in each frame along with additional data such as the frequency offset, the frame length recovered, and more.



```
Terminal
File  Edit  View  Search  Terminal  Help

----------------------------------------------------------------
Tag Debug: Rx Bytes
Input Stream: 00
  Offset: 86304  Source: n/a     Key: packet_num   Value: 901
  Offset: 86304  Source: n/a     Key: rx_time   Value: {43 0.2515}
  Offset: 86304  Source: n/a     Key: ofdm_sync_carr_offset   Value: 0
  Offset: 86304  Source: n/a     Key: ofdm_sync_chan_taps   Value: #[(0,0) (0,0) (0,0) (0,0) (0,0) (0,0) (-57.3588,28.0875) (-27.5464
,57.9232) (17.3605,61.5302) (51.9943,37.698) (64.1429,-3.44512) (47.2486,-42.6445) (7.77717,-62.493) (-32.9584,-55.1198) (-60.9536,-2
0.6291) (-58.6457,24.0368) (-31.8996,55.2848) (9.93802,63.237) (47.9724,41.9092) (64.3968,1.84674) (51.0482,-38.6227) (14.1862,-62.82
17) (-29.4563,-57.0344) (-6.25999,-3.50976) (-6.21481,1.68404) (-37.8103,51.6988) (0.300568,6.94904) (5.65531,5.5312) (7.25981,2.0291
) (5.77555,-3.50003) (2.51427,-6.35243) (-3.19984,-6.49531) (0,0) (-6.72901,1.56252) (-4.52989,5.06416) (0.114071,7.13498) (4.03097,6
.6017) (6.93602,1.31269) (6.18289,-2.83796) (28.0812,-57.3779) (-1.66975,-6.46062) (-6.09075,-3.72129) (-7.04198,0.748462) (-5.32309,
5.65412) (-0.994682,7.01674) (3.59861,5.9754) (6.55612,1.97089) (6.45307,-2.36795) (3.92974,-5.75586) (-1.09591,-6.65997) (-5.23108,-
4.51511) (-6.57645,-0.287068) (-5.67794,4.32635) (-15.7438,61.7683) (2.89212,5.94307) (6.23361,2.5434) (6.75737,-1.76117) (4.07846,-5
.47053) (-0.847337,-7.02899) (0,0) (0,0) (0,0) (0,0) (0,0)]
  Offset: 86304  Source: n/a     Key: packet_len   Value: 96
  Offset: 86400  Source: n/a     Key: packet_num   Value: 902
  Offset: 86400  Source: n/a     Key: rx_time   Value: {43 0.2995}
  Offset: 86400  Source: n/a     Key: ofdm_sync_carr_offset   Value: 0
  Offset: 86400  Source: n/a     Key: ofdm_sync_chan_taps   Value: #[(0,0) (0,0) (0,0) (0,0) (0,0) (0,0) (-52.7853,36.0452) (-18.6584
,61.8289) (24.97,59.5648) (56.9949,30.0243) (61.6248,-13.0398) (46.9443,-44.5984) (-1.56134,-64.3712) (-40.7092,-49.0412) (-62.5502,-
12.2038) (-56.3453,30.0715) (-23.6562,59.7398) (17.8139,60.9349) (52.6579,35.4681) (63.4275,-6.31681) (45.1697,-45.4104) (6.25198,-63
.8661) (-35.0172,-53.144) (-6.53113,-1.79424) (-6.59392,2.39228) (-35.49,52.5215) (1.3804,6.84095) (5.21699,4.20559) (6.97569,0.35925
) (6.51963,-3.97133) (1.52416,-6.72884) (-4.0989,-7.0354) (0,0) (-6.75466,1.62282) (-3.83334,5.45942) (0.620617,6.54834) (4.57446,4.8
7313) (7.48565,1.04591) (5.98766,-3.68139) (25.4957,-58.3084) (-2.31596,-7.09009) (-5.84614,-3.02347) (-6.37633,1.36172) (-4.15532,5.
12278) (0.235253,7.02369) (4.19478,5.15179) (6.683,1.49791) (6.75875,-3.90718) (3.00587,-6.5385) (-2.34179,-6.6501) (-6.20391,-4.0419
5) (-7.10068,0.682619) (-4.66995,4.99666) (-13.1168,62.8357) (3.68903,5.78655) (6.60382,2.5861) (6.9354,-3.34732) (3.96451,-5.8993) (
-1.46468,-6.64297) (0,0) (0,0) (0,0) (0,0) (0,0)]
  Offset: 86400  Source: n/a     Key: packet_len   Value: 96
----------------------------------------------------------------
she said to herself, and began by taking the little golden key,
and unlocking the door that led into the garden.  Then she went

----------------------------------------------------------------
Tag Debug: Rx Bytes
Input Stream: 00
  Offset: 86496  Source: n/a     Key: packet_num   Value: 903
  Offset: 86496  Source: n/a     Key: rx_time   Value: {43 0.3475}
  Offset: 86496  Source: n/a     Key: ofdm_sync_carr_offset   Value: 0
  Offset: 86496  Source: n/a     Key: ofdm_sync_chan_taps   Value: #[(0,0) (0,0) (0,0) (0,0) (0,0) (0,0) (-57.8425,28.0773) (-27.4106
,56.9704) (14.393,62.5558) (51.235,38.8964) (63.4193,-2.1136) (40.2063,-48.7542) (9.00743,-63.6887) (-32.7648,-54.8907) (-59.3145,-21
.2349) (-61.0457,21.0678) (-32.7136,53.1675) (9.64152,62.9865) (47.9198,42.2373) (64.8589,2.14258) (51.2807,-37.4272) (15.5905,-62.10
6) (-27.4064,-57.9648) (-6.3465,-3.06762) (-6.77371,1.80122) (-30.2927,56.3321) (-0.044856,6.89226) (4.94646,5.66306) (6.57593,0.9927
91) (6.2097,-3.87359) (1.89169,-6.75676) (-1.91067,-7.41739) (0,0) (-6.8577,1.04385) (-4.54442,5.23095) (-0.0070786,6.71393) (4.45323
,5.82931) (6.99094,1.64243) (6.2546,-4.02868) (19.356,-61.3303) (-1.62783,-7.23274) (-5.3427,-3.78643) (-6.57256,0.424797) (-5.27096,
4.32151) (-0.973295,6.47935) (3.54353,5.85222) (7.26021,1.66533) (6.55015,-2.22822) (3.27198,-5.8465) (-0.663574,-7.15649) (-5.44069,
-4.255) (-7.54408,-0.690403) (-5.19024,3.99966) (-7.102,62.0971) (2.55996,6.74378) (5.89858,2.78679) (6.6499,-1.60175) (3.85084,-6.09
336) (-0.644496,-6.96418) (0,0) (0,0) (0,0) (0,0) (0,0)]
  Offset: 86496  Source: n/a     Key: packet_len   Value: 96
----------------------------------------------------------------
```

Experiment Questions -

1. Calculate the signal's coherence bandwidth and coherence time and deduct the maximal possible parameters that we can feed the frequency selective fading channel model module. Prove your calculation using the OFDM system and changing the Power Delay Profile.

   (At which channel conditions the communication should fail).

2. Repeat using the Rician/LOS (instead of Rayleigh/NLOS). **Explain** the difference.

3. In our experiment the frame length is 96, Use the defined Variables (like FFT length, number of carriers, and more) to calculate how many characters are transferred in each frame. Derive an expression with these variables that dictate the frame length. Show your calculations and **explain.**

4. **Bonus 30%:** Change the OFDM system parameters to work in even harder frequency selective fading channel model conditions. Show proof of your work and **explain** what was changed to achieve this.

   (Changing the sample rate does not change the system performance in this case)

Good luck!