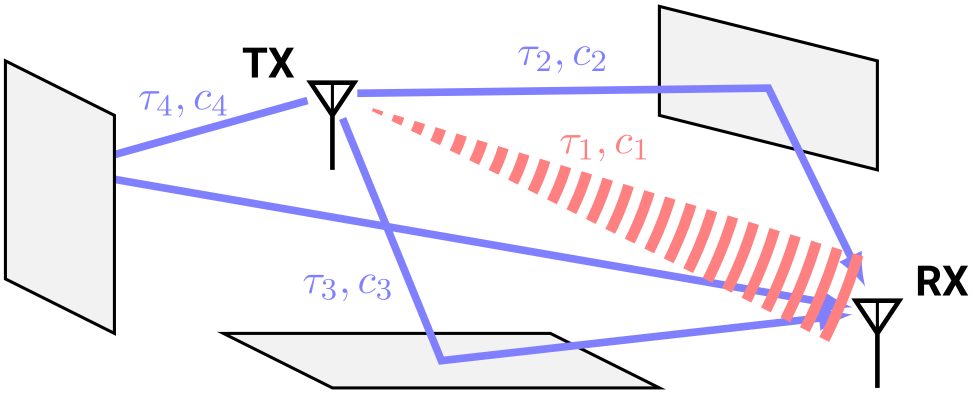
Multipath Fading Demonstration Platform using Software Defined Radio

Naoki Sean Pross, *Sara Cinzia Halter*

Semester Thesis
OST — Eastern Switzerland University of Applied Sciences, Rapperswil
December 23$^{\text{rd}}$, Fall Semster 2021



Field of Study	Wireless Communications
Topic	Multipath Fading
Examiner	Prof. Dr. Heinz Mathis

Abstract

Today wireless devices are omnipresent and have to work under technically adverse circumstances. Among the multitude of problems affecting wireless communication systems, a common issue is what is known as the *multipath fading effect*. The aim of this work is to create a demonstrative platform that can illustrate how multipath fading degrades a wireless communication link. A number of scenarios affected by multipath fading were elaborated, simulated and measured using two USRP B210 software defined radios and the GNU Radio signal processing toolkit. To demonstrate the effects, a custom graphical front-end for GNU Radio was built using the Dear IMGUI framework.

Contents

List of symbols

Mathematics	
j	Imaginary unit
ϕ^*	Complex conjugate of ϕ
$\mathbf{v}$, $\mathbf{m}$	Vector quantity
Physical quantities or constants	
t	Time
T	Period
τ	Convolution time
f	Frequency in Hz
ω	Frequency in radians per second
Ω	Digital frequency in radians per sample
φ	Phase angle
c_0	Speed of light
v	Velocity
Modulation schemes	
$n(t)$	Additive white Gaussian noise (AWGN)
$f(t)$	Multiplicative fading noise
$p(t)$	Pulse function
$m(n)$	Message samples
$m(t)$	Message waveform
$\phi(t)$	Carrier signal
$s(t)$	Signal waveform
$s(n)$	Signal samples
$r(t)$	Received signal
$r(m)$	Received samples
Channel modelling	
$h(t)$	LTI impluse response, $h(\tau, t)$ for the LTV impulse response
$h(m)$	LTI discrete time impulse response, $h_l(m)$ for the LTV variant
$H(f)$	LTI Frequency response, $H(f, t)$ for the LTV frequency response
c_k	Tap attenuation
τ_k	Tap delay
T_d	Delay spread
B_c	Coherence bandwidth
R_{xy}	Correlation between x and y, also written as $x \star y$

List of Acronyms

Acronym	Description
API	Application Programming Interface
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
CAZAC	Constant Amplitude Zero Autocorrelation waveform
CIR	Channel Impulse Response
CMA	Constant Modulus Adaptive filter
DPG	Dear PyGUI
ETSI	European Telecommunication Standards Insitite
FIR	Finite Impulse Response
GR	GNU Radio
GRC	GNU Radio Companion
GSM	Global System for Mobile communication
IMGUI	Immediate Mode Graphical User Interface
IP	Internet Protocol
IoT	Internet of Things
LMS-DD	Least Mean Square Decision-Directed equalizer
LOS	Line Of Sight
LTI	Linear Time Invariant
LTV	Linear Time-Varying
MSB	Most Significant Bit
NLOS	No Line Of Sight
PDP	Power Delay Profile
PSAM	Pilot Symbols Assisted Modulation
PSK	Phase Shift Keying
QAM	Quadrature Amplitude Modulation
QPSK	Quaternary Phase Shift Keying
SDR	Software Defined Radio
UDP	User Datagram Protocol
US	Uncorrelated Scattering
WSS	Wide Sense Stationary
WSSUS	Wide Sense Stationary Uncorrelated Scattering

1 Introduction

1.1 Background

It is undeniable that in the last two decades wireless devices have become extremely ubiquitous, and are no longer employed under carefully chosen conditions.

Nowadays smart phones, internet of things (IoT) and many other wireless applications are omnipresent and have to work in environments that are very far from ideal. Furthermore, in addition to the already large class of networked appliances, the next generation of wireless devices in urban environments will include the new category of vehicles [5]. While at the same time in rural regions, developing countries as well as in other low-user density areas, wireless transmission links using mesh networks have become a practical alternative to wired broadband [2, 14, 18]. Thus today the study of problems concerning wireless devices is a very relevant topic.

In particular, a common issue observed in the previously mentioned use cases is the so called *multipath fading effect*, that degrades the reliability of a wireless transmission link [11, 10]. The problem of fading was actually foreseen [4, 15] and today most modern transmission schemes implement measures to reduce the effects fading [11, 12].

This work studies the multipath fading effect, and how it affects modern digital transmission systems that use quadrature amplitude (QAM) and phase shift keying (PSK) modulation.

1.2 Task description

As described in the document given at the beginning of the semester:

> The goal is to develop a SDR-based demonstrator, consisting of one transmitter and one receiver, to illustrate the impact of different fading effects on the signal. To get a brief understanding of the concept of fading channels, the project should be started with a literature research followed by simulation of different scenarios, which then can be reproduced by measurements.

The entire task description is found in the appendix.

1.3 Overview

In chapter 2, the theoretical formulations and mathematical basis necessary to understand the problem analyzed by this work are presented. Specifically M-ary QAM, M-PSK modulation, and three models for multipath fading (continuous time, discrete time and statistical) are explained. Chapter 3 describes in detail our implementation. The transmitter and receiver chains are explained, and simulations as well as measurements under different multipath fading conditions are presented. Finally some problems of the current implementation are addressed. Chapter 4 discusses the results of the project, and suggests how the device could be improved in the future.

2 Theory

2.1 Overview

The sections 2.2 and 2.3 will briefly introduce the mathematical formulations of the QAM and PSK modulation schemes. Section 2.4 the channel models used in this project are explained. The setup is summarized as a block diagram in Fig. 2.1. For conciseness, encoding schemes and (digital) signal processing calculations are left out and discussed later. Section 2.4 presents an established mathematical model to understand multipath fading, as well as a brief description of a discrete-time model and the intricacies caused by the sampling process. Finally, the concept of stochastic models is mentioned, as they are often used to simulate multipath channels [9, 11].

2.2 Quadrature amplitude modulation (M-ary QAM)

Quadrature amplitude modulation is a family of modern digital modulation methods, that use an analog carrier signal. The simple yet effective idea behind QAM is to encode extra information into an orthogonal carrier signal, thus increasing the number of bits sent per unit of time (symbol) [10, 19, 11, 12]. A block diagram of the process is shown in Fig. 2.2.

2.2.1 Modulation of a digital message

Bit splitter As mentioned above, quadrature modulation allows to send more than one bit per unit time. The first step of the process, is to use a so called bit splitter, that converts the continuous bit stream $m(n)$ into pairs of chunks of $\kappa = \log_2 \sqrt{M}$ bits each, where M is a power of 2. The two bit vectors of length κ, denoted by $\mathbf{m}_i$ and $\mathbf{m}_q$ in figure 2.2, are called in-phase and quadrature component respectively[12]. The reason will become more clear later.

Binary to level converter Both bit vectors $\mathbf{m}_i, \mathbf{m}_q \in \{0,1\}^\kappa$ are sent through a binary to level converter. It's purpose is to reinterpret the bit vectors as numbers, usually in gray code, and to convert them into analog waveforms, which we will denote with $m_i(t)$ and $m_q(t)$ respectively. Mathematically the binary to level converter can be described as:

$$m_i(t) = \text{Level}(\mathbf{m}_i) \cdot p(t), \tag{2.1}$$

i.e. a pulse function[1] $p(t)$ scaled by the interpreted binary value, written here using a "Level" function. So there are $2^\kappa = \sqrt{M}$ levels the analog waveforms $m_i(t)$ and $m_q(t)$ can take, and each level encodes κ bits per unit time.

[1] Typically a root raised cosine to optimize for bandwidth [12].

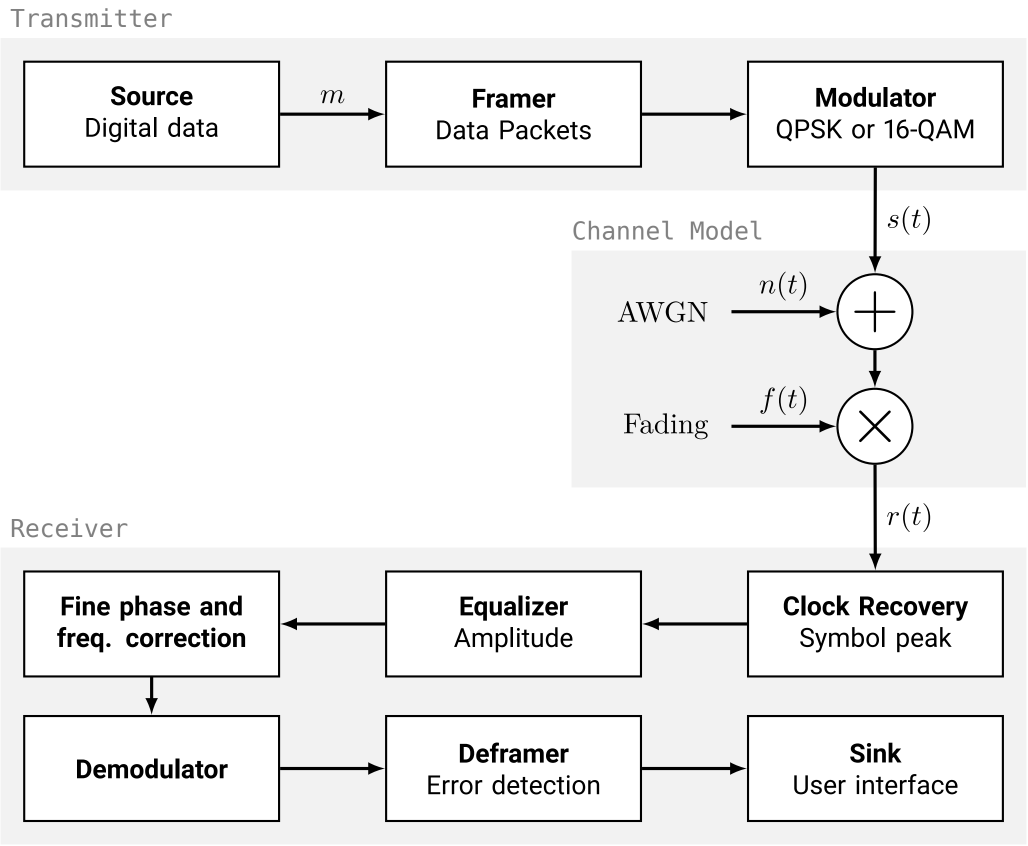


Figure 2.1 Block diagram of the developed wireless communication system with annotated signal names. Frequency domain representations of signals use the uppercase symbol of their respective time domain name. Amplification constants in the channel (for $s(t)$ and $r(t)$) were omitted throughout the document for readability.

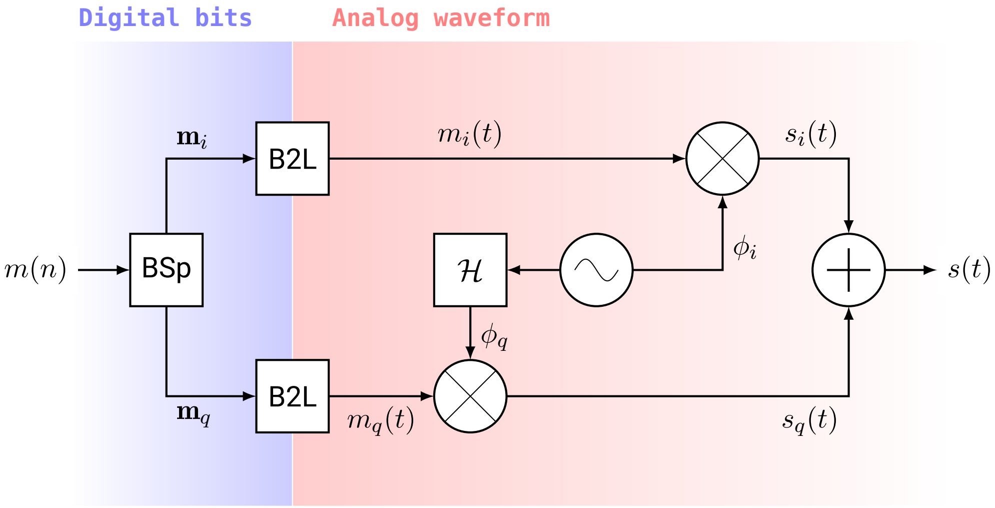


Figure 2.2 Block diagram of a M-ary QAM modulator.

Mixer By having analog level signals, it is now possible to mix them with radio frequency carriers. Because there are two waveforms, one might expect that two carrier frequencies are necessary, however this is not the case. The two component $m_i(t)$ and $m_q(t)$ are mixed with two different periodic signals $\phi_i(t)$ and $\phi_q(t)$ that have the same frequency $\omega_c = 2\pi/T$. How this is possible is explained in the next section.

2.2.2 Orthogonality of carrier signals

Before explaining how the two carrier signals are generated, some important mathematical properties of ϕ_i and ϕ_q have to be discussed. In order to modulate two messages over the same frequency ω_c, the two carriers need to be *orthonormal*[2] to each other. Mathematically this is expressed by the conditions [10]

$$\langle \phi_i, \phi_q \rangle = \int_T \phi_i \phi_q^* \, dt = 0, \text{ and} \tag{2.2a}$$

$$\langle \phi_k, \phi_k \rangle = \int_T \phi_k \phi_k^* \, dt = 1, \text{ where } k \text{ is either } i \text{ or } q. \tag{2.2b}$$

Provided these rather abstract conditions, let's define a new signal

$$s = m_i \phi_i + m_q \phi_q. \tag{2.3}$$

Notice that assuming m_i and m_q are constant[3] over the carrier's period T,

$$\begin{aligned} \langle s, \phi_i \rangle = \int_T s\phi_i^* \, dt &= \int m_i \phi_i \phi_i^* + m_q \phi_q \phi_i^* \, dt \\ &= m_i \underbrace{\int_T \phi_i \phi_i^* \, dt}_{1} + m_q \underbrace{\int_T \phi_q \phi_i^* \, dt}_{0} = m_i, \end{aligned}$$

which effectively means that it is possible to isolate a single component $m_i(t)$ out of $s(t)$. The same of course works with ϕ_q as well resulting in $\langle s, \phi_q \rangle = m_q$. Thus (remarkably) it is possible to send two different signals over the same frequency, without them interfering with each other. Since each signal can represent one of $\sqrt{M}$ values, by having two we obtain $\sqrt{M} \cdot \sqrt{M} = M$ possible combinations.

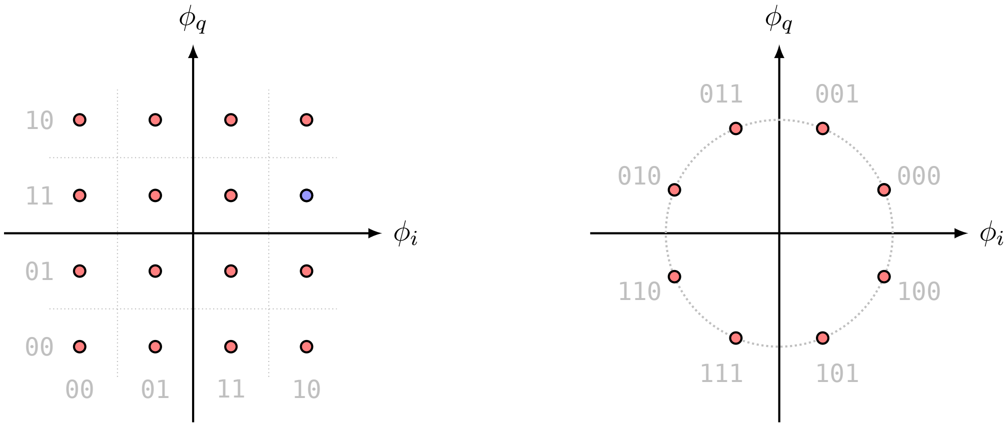
A graphical way to see what is happening, is to observe a so called *constellation diagram.* An example is shown in Fig. 2.3(a) for $M = 16$. The two carrier signals ϕ_i and ϕ_q can be understood as bases of a coordinate system, in which the two amplitude levels of the modulated messages determine a position in the grid.

Example A concrete example for $M = 16$: if the message is 1110 the bit splitter creates two values $\mathbf{m}_q = 11$ and $\mathbf{m}_i = 10$; both are converted into analog amplitudes (symbols) $m_q = 3$ and $m_i = 4$; that are then mixed with their respective carrier, resulting in $s(t)$ being the point inside the bottom right sub-quadrant of the top right quadrant (blue dot in Fig. 2.3(a)).

In Fig. 2.3(a) the dots of the constellation have coordinates that begin on the bottom left corner, and are nicely aligned on a grid. Both are not a necessary requirement for QAM, in

[2] Actually orthogonality alone would be sufficient, however then the left side of (2.2b) would not equal 1, and an inconvenient factor would be introduced in many later equations [10, 12].

[3] This is an approximation assuming that the signal changes much slower relative to the carrier.



(a) 16-QAM **(b)** 8-PSK

Figure 2.3 Examples of constellation diagrams. Each dot represents a possible location for the complex amplitude of the passband signal.

fact there are schemes that are arranged on a non square shape, and place the dots in different orders. For example, when $M = 32$ the dots are arranged on a 6×6 grid, but the outermost corners are not used. The only constraint that most QAM modulators have in common, with regards to the geometry of the constellation, is that between any two adjacent dots (along the axis, not diagonally) only one bit of the represented value changes (gray code). This is done to improve the bit error rate (BER) of the transmission.

2.2.3 Construction of orthogonal carrier signals

Knowing why there is a need for orthogonal carriers, it should now be discussed which functions satisfy the property described by (2.2a). If ϕ_i is a real valued signal (which is typical) it is possible to find a function for the quadrature carrier using the *Hilbert transform* (sometimes called Hilbert filter):

$$\mathcal{H}\, g(t) = g(t) * \frac{1}{\pi t} = \frac{1}{\pi} \int_{\mathbb{R}} \frac{g(\tau)}{t-\tau}\, d\tau = \frac{1}{\pi} \int_{\mathbb{R}} \frac{g(t-\tau)}{\tau}\, d\tau. \tag{2.4}$$

The Hilbert transform is a linear operator that introduces a phase shift of $\pi/2$ over all frequencies [12, 10], and it is possible to show that given a real valued function $g(t)$ then $\langle g, \mathcal{H}\, g \rangle = 0$ [13, 19]. There are many functions that are Hilbert transform pairs, however in practice the pair $\phi_i(t) = \cos(\omega_c t)$ and $\phi_q(t) = \mathcal{H}\, \phi_i(t) = \sin(\omega_c t)$ is always used.

2.3 Phase shift keying (M-PSK)

Phase shift keying (PSK) is another popular family of modulation schemes for digital signals that is simpler than QAM. In PSK as the name suggests only the phase of the envelope changes, which implies that all symbols have the same amplitude. Thus, instead of arranging the symbols into a grid, as is done in QAM, M-PSK distributes the symbols over the unit circle at equidistant intervals of $2\pi/M$ radians [11, 19]. An example of 8-PSK is shown in Fig. 2.3(b). Mathematically the process of a PSK modulation can be described by making

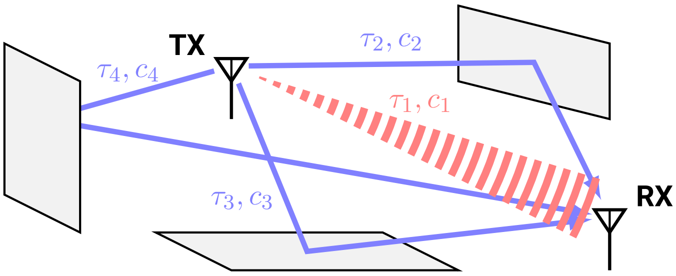


Figure 2.4 Sketch of channel with multipath fading.

the carrier's phase of the message. For a complex exponential carrier:

$$s(t) = \exp j\left(\omega_c t + \varphi(t)\right), \quad \text{where} \quad \varphi = \frac{2\pi \cdot \text{Level}(\mathbf{m})}{M}, \quad \mathbf{m} \in \{0,1\}^{\log_2 M}. \tag{2.5}$$

It is worth noting that the case of 4-PSK, also known as quaternary phase shift keying (QPSK), is a special case, because its constellation is (up to a constant phase difference) the same as a 4-ary QAM.

2.4 Multipath fading

In the previous section, we discussed how the data is modulated and demodulated at the two ends of the transmission system. This section discusses what happens between the transmitter and receiver when the modulated passband signal is transmitted wirelessly.

In theory because wireless transmission happens through electromagnetic radiation, to model a wireless channel one would need to solve Maxwell's equations for either the electric or magnetic field, however in practice that is usually not (analytically) possible. Instead what is typically done, is to model the impulse response of the channel using a geometrical or statistical model, that is parametrized by a set of coefficients, which are either simulated or measured experimentally [10].

In our relatively simple model we are going to include an additive white Gaussian noise (AWGN) and a Rician (or Rayleighan) fading; both are required to model physical effects of the real world. The former in particular is relevant today, as it describes the noise patterns of dense urban environments [9].

2.4.1 Geometric model

The simplest way to understand multipath fading, is to consider it from a geometrical perspective. Fig. 2.4 is a sketch of a wireless transmission system affected by multipath fading. The transmitter's antenna radiates an electromagnetic wave in the direction of the receiver (red line), however even under the best circumstances a part of the signal is dispersed in other directions (blue lines).

The problem is that, as is geometrically evident, some paths are longer than others. Because of electromagnetic wave travel at a constant speed, the signal is seen by the receiver multiple times with different phase shifts [10, 9]. To analytically model this effect, we describe the received signal $r(t)$ as a linear combination of delayed copies of the sent signal $s(t)$, each with

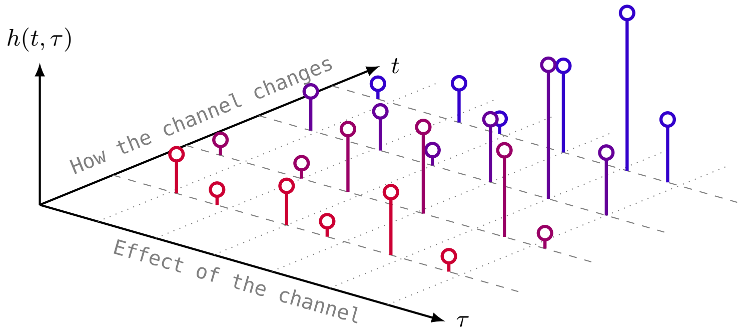


Figure 2.5 LTV impulse response of a multipath fading channel.

a different attenuation c_k and phase shift τ_k:

$$r(t) = \sum_k c_k s(t - \tau_k). \tag{2.6}$$

The linearity of the model is justified by the assumption that the underlying electromagnetic waves behave linearly (superposition holds) [10]. How many copies of $s(t)$ (usually referred to as "taps" or "rays") should be included in (2.6), depends on the precision requirements of the model.

A further complication arises, when one end (or both) is not stationary. In that case the lengths of the paths change over time, and as a result the delays τ_k and the attenuations c_k become functions of time: $\tau_k(t)$ and $c_k(t)$ respectively [10, 9]. Even worse is when the velocity at which the device is moving is high, since then Doppler shifts of the electromagnetic wave frequency become non negligible [10].

Thus the arrangement can be modelled as a linear time-*varying* system (LTV), if the transmitter or the receiver (or anything else in the channel) is moving, and as a linear time *invariant* (LTI) system if the geometry is stationary. We can rewrite an LTV version of equation (2.6) using a convolution product as follows:

$$\begin{aligned} r(t) = \sum_k c_k(t) s(t - \tau_k(t)) &= \sum_k c_k(t) \int_{\mathbb{R}} s(\tau)\delta(\tau - \tau_k(t))\, d\tau \\ &= \int_{\mathbb{R}} s(\tau) \sum_k c_k(t)\delta(\tau - \tau_k(t))\, d\tau = s(\tau) * h(\tau, t), \end{aligned}$$

obtaining a new function

$$h(\tau, t) = \sum_k c_k(t)\delta(\tau - \tau_k(t)), \tag{2.7}$$

that describes the *channel impulse response* (CIR). This function depends on two time parameters: actual time t and convolution time τ. To better understand $h(\tau, t)$, consider an example shown in Fig. 2.5. Each stem represents a weighted Dirac delta, so each series of stems of the same color, along the convolution time τ axis, is a channel response at some specific time t. Along the other axis we see how the entire channel response changes over time[4]. Notice that the stems are not quite aligned to the τ time raster (dotted lines), that is because in (2.7) not only the weights c_k but also the delays τ_k are time dependent.

[4] In the figure only a finite number of stems was drawn, but actually the weights $c_k(t)$ of the Dirac deltas change continuously.

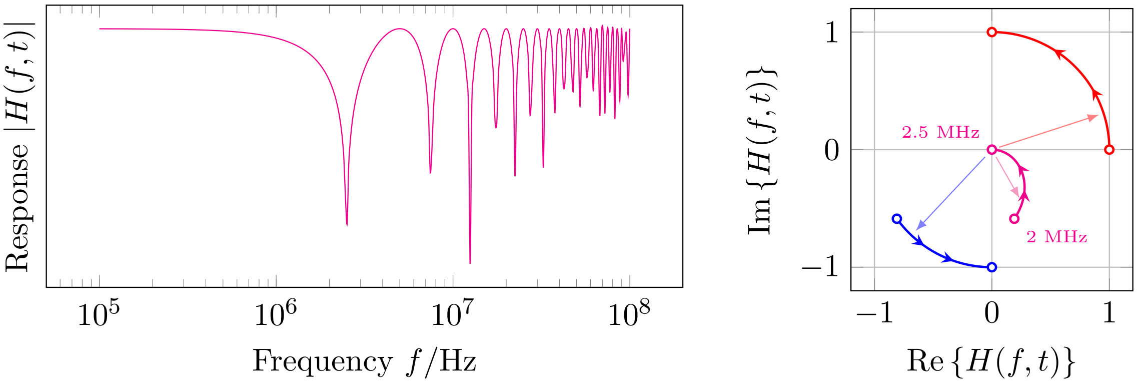


Figure 2.6 Frequency response of a multipath fading channel.

2.4.2 Spectrum of a multipath fading channel

With a continuous time channel model the spectral properties of a fading channel can now be discussed. The frequency response is the Fourier transform of the impulse response, i.e. $H(f,t) = \mathcal{F}\,h(\tau,t)$, though in this case $h(\tau,t)$ depends on two time variables, but that is actually not an issue. It just means that the frequency response is also changing over time. Hence we perform the Fourier transform with respect to the channel (convolution) time variable τ to obtain

$$H(f,t) = \int_{\mathbb{R}} \sum_k c_k(t)\delta(\tau - \tau_k(t))e^{-2\pi j f\tau}\, d\tau = \sum_k c_k(t)e^{-2\pi j f \tau_k(t)}. \tag{2.8}$$

Equation (2.8) shows that the frequency response is a periodic complex exponential. This has some important implications. Notice that if there is only one tap (term), the magnitude of $H(f,t)$ is a constant (with respect to f) since $|e^{j\alpha f}| = 1$. This means that the channel attenuates all frequencies by the same amount, therefore it is said to be a *frequency non-selective* or *flat fading* channel. Whereas in the case when there is more than one tap, the taps interfere destructively at certain frequencies and the channel is called *frequency selective.* To illustrate how this happens, plots of the frequency response of a two tap channel model are shown in Fig. 2.6. On the left is the magnitude of $H(f,t)$, which presents periodic “dips” when the taps interfere destructively. On the right complex loci for the two taps (red and blue), as well as their sum (magenta) are shown, for values over the frequency range near the first dip (2 to 2.5 MHz).

2.4.3 Quantifying dispersion

Having discussed how multipath fading affects communication systems, the next important step is to be able to quantify its effects in order to to compare different multipath channels to each other.

An intuitive parameter to quantify how dispersive a channel is, is to take the time difference between the fastest and slowest paths with significant energy. In the literature this is called *delay spread* [9], and here it is denoted by T_d. Consequently, a low delay spread means that all paths have more or less the same length, while a high delay spread implies that there is a large difference in length among the paths. Thus T_d could be be defined as

$$T_d = \max_k(\tau_k(t)) - \min_k(\tau_k(t)), \tag{2.9}$$

as is done in [10]. However since in reality some paths get more attenuated than others ($c_k(t)$ parameters) it also not uncommon to define the delay spread as a weighted mean or even as a statistical second moment (RMS value), where mean tap power $\mathrm{E}\{|c_k(t)|^2\}$ is taken into account [11, 9].

Another important parameter for quantifying dispersion is *coherence bandwidth*, a measure that is highly related to delay spread but in the frequency domain. Coherence bandwidth, is informally "how much bandwidth can be used by the signal before it gets distorted" (in our case by fading) [9]. Thus intuitively, this parameter must be related to the delay spread with an inversely proportional relationship, since higher delay spread implies more intersymbol interference. And in fact, although there are multiple definitions depending on the context, the coherence bandwidth B_c is usually estimated with

$$B_c \approx \frac{1}{T_d}. \tag{2.10}$$

Finally, another important mean of parametrizing a multipath fading channel is what is called a *power delay profile* (PDP). PDPs are nothing but a list of taps for a FIR model of multipath fading [11]. The weight of each tap in the PDP corresponds to the average channel tap power (hence the name *power* delay profile) and is usually given in decibel [11, 9]. An example is shown at the end of chapter 3 in Tab. 3.2.

2.4.4 Discrete-time model

Since in practice signal processing is done digitally, it is meaningful to discuss the properties of a discrete-time model. To keep the complexity of the model manageable some assumptions are necessary, thus the sent discrete signal[5] $s(n)$ is assumed to have a finite single sided bandwidth W and the sampling is assumed to be ideal. This implies that the time-domain signal is a series of sinc-shaped pulses each shifted from the previous by a time interval $T = 1/(2W)$ (Nyquist rate) [9]:

$$s(t) = \sum_n s(n) \operatorname{sinc}\left(\frac{t}{T} - n\right). \tag{2.11}$$

The waveform $s(t)$ is then convolved with the CIR function $h(\tau, t)$ (with respect to τ) from the continuous time model resulting in the waveform at the receiver

$$\begin{aligned} r(t) &= \int_{\mathbb{R}} \sum_n s(n) \operatorname{sinc}\left(\frac{\tau}{T} - n\right) \sum_k c_k(t) \delta(\tau - \tau_k(t))\, d\tau \\ &= \sum_n s(n) \sum_k c_k(t) \operatorname{sinc}\left(\frac{t - \tau_k(t)}{T} - n\right), \end{aligned}$$

which is then sampled at the Nyquist rate of $2W = 1/T$, resulting in a set of samples[6] [9]:

$$r(m) = \sum_n s(n) \sum_k c_k(mT) \operatorname{sinc}\left(m - \frac{\tau_k(mT)}{T} - n\right).$$

[5] This is an abuse of notation. The argument n is used to mean the n-th digital sample of s, whereas $s(t)$ is used for the analog waveform. A more correct but longer notation is $s(nT)$, where T is the sample time.

[6] Again, the notation $r(m)$ means the m-th digital sample of $r(t)$, i.e. $r(mT)$.

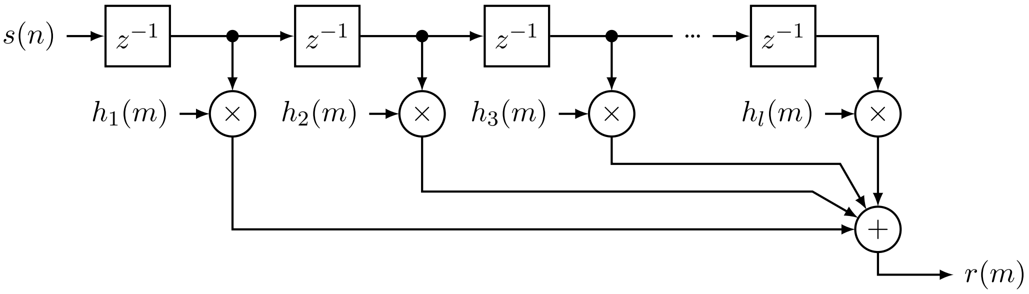


Figure 2.7 Fading channel as a tapped delay line.

Finally, the substitution $l = m - n$ eliminates the sender's sample counter n (unknown to the receiver), and reformulates $r(m)$ as a discrete convolution product of $s(n)$ with a discrete CIR function $h_l(m)$ [9]:

$$r(m) = \sum_l s(m-l) \sum_k c_k(mT) \operatorname{sinc}\left(l - \frac{\tau_k(mT)}{T}\right) = \sum_l s(m-l) h_l(m). \tag{2.12}$$

This result is very similar to the continuous time model described by (2.7) in the sense that each received digital sample is a sent sample convolved with a different discrete channel response (because of time variance). To see how the discrete CIR

$$h_l(m) = \sum_k c_k(mT) \operatorname{sinc}\left(l - \frac{\tau_k(mT)}{T}\right) \tag{2.13}$$

is different from (2.7) consider again the plot of $h(\tau, t)$ in Fig. 2.5. The plot of $h_l(m)$ would have discrete axes with m replacing t and l instead of τ. Because of the finite bandwidth in the l axis instead of Dirac deltas there would be superposed sinc functions.

From a signal processing perspective (2.13) can be interpreted as a simple tapped delay line [9, 10], as schematically drawn in Fig. 2.7. Together with linearity, this confirms that the presented model is indeed just a FIR filter. Simple multipath channels can be simulated with just a few lines of code, for example the data for the static fading channel in Fig. 2.6 is generated in just four lines of Python. The difficulty of fading channels in practice lies in the estimation of the constantly changing parameters $c_k(t)$ and $\tau_k(t)$ [9].

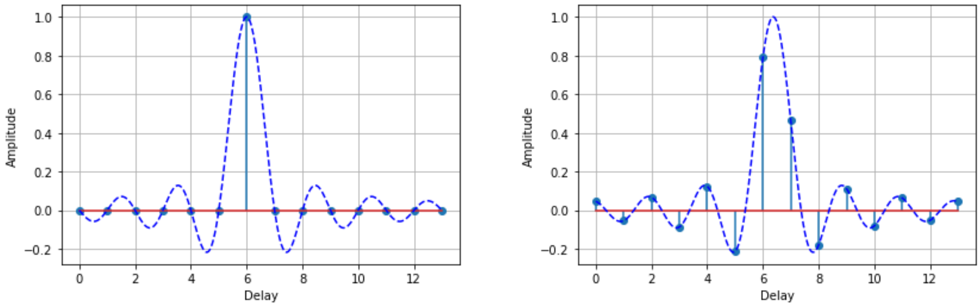
2.4.5 Simulating multipath CIR with FIR filters

As mentioned in the previous section a FIR filter can be used to simulate discrete-time models of multipath fading. But with FIR filters the delays can only be integer multiples of the sample rate. When the delays are non integer an approximation needs to be done. That is because FIR filters have a transfer function of the form

$$H(j\omega) = \sum_{n=0}^{N} h(n) e^{-j\omega nT} \quad \text{commonly written as} \quad H(z) = \sum_{n=0}^{N} h(n) z^{-n}, \tag{2.14}$$

but a non integer delay of τ in the frequency domain is $H_\tau(j\omega) = e^{-j\omega\tau}$. There are multiple ways to find coefficients $h(n)$ that approximate H_τ [21]. In this case the least squares method was used by minimizing the error function

$$E(j\omega) = H(j\omega) - H_\tau(j\omega). \tag{2.15}$$



(a) Integer delay of 6 samples.

(b) Fractional delay of 6.37 samples.

Figure 2.8 FIR filters for integer and fractional delays.

The least square method together with the assumption of finite bandwidth and the requirement of causality gives the following rule for computing the FIR filter coefficients [21]:

$$h(n) = \begin{cases} \operatorname{sinc}(n-\tau) & 0 \leq n \leq N \\ 0 & \text{otherwise,} \end{cases} \tag{2.16}$$

where the odd order of the filter N should satisfy the condition

$$N = 2\lfloor \tau \rfloor + 1 \tag{2.17}$$

for a minimal error in the approximation [21]. It is worth mentioning that it is also possible to build FIR filters of even length with a different condition, or ones that do not satisfy (2.17), in which cases more consideration is required. An example of a fractional delay FIR filter is shown in Fig. 2.8. Where the sinc function does not select an integer sample, which in turn means that the other sampled values do not add up to zero. Thus, they will be distributed among the other whole numbers.

2.4.6 Statistical model

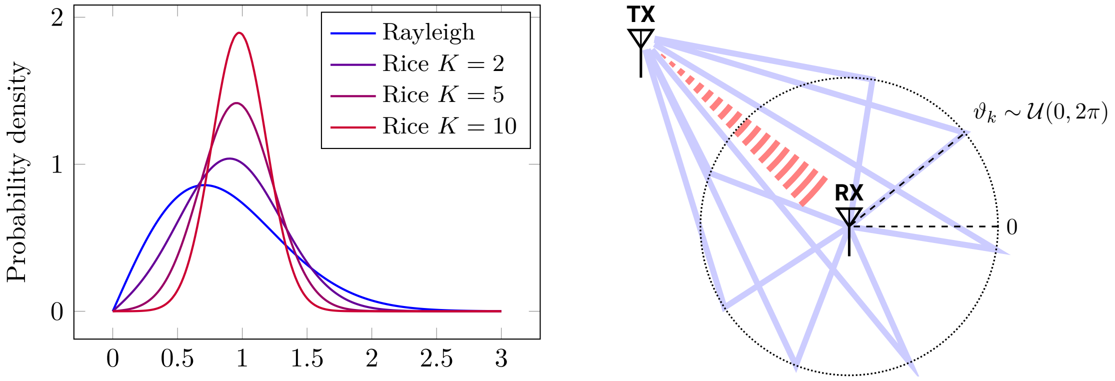
Because as mentioned earlier it is difficult to estimate the time-dependent parameters of $h_l(m)$ in many cases it is easier to model the components of the CIR as stochastic processes, thus greatly reducing the number of parameters [9, 11]. This is especially effective for channels that are constantly changing, because by the central limit theorem the cumulative effect of many small changes tends to a normal distribution.

Before discussing the models themselves, their underlying statistical assumptions need to be considered. In the literature the so called WSSUS assumptions are made [9, 10], which for a LTV CIR $h(\tau, t)$ can be formulated as

$$R(\tau') = \mathrm{E}\left\{h(\tau,t)h^*(\tau+\tau',t)\right\}, \text{ and} \tag{2.18a}$$

$$0 = \mathrm{E}\left\{h(\tau,t)h^*(\tau,t')\right\} \text{ for } t \neq t'. \tag{2.18b}$$

Equation (2.18a) states that the fading CIR is a *wide sense stationary* (WSS) stochastic process, while (2.18b) is the *uncorrelated scattering* assumption, which loosely speaking states that the paths do not interfere with each other. The latter is more realistic than the former, but WSS is still useful as it considerably simplifies the mathematical formulation [9].



(a) Amplitude density $p(a)$.

(b) Ring of scattering objects.

Figure 2.9 Statistical model for multipath fading.

NLOS case Recall that $h(\tau, t)$ is a function of time because c_k and τ_k change over time. The idea of the statistical model is to replace the cumulative change caused by c_k and τ_k (which are difficult to estimate) with a single random variable f. This is done as follows.

Multipath fading is a form of multiplicative noise, as confirmed by the fact that convolving a complex baseband signal $e^{j\omega_c t}$ with the fading CIR $h(\tau, t)$ gives

$$e^{j\omega_c\tau} * h(t,\tau) = \sum_k c_k(t) e^{j\omega_c(\tau-\tau_k(t))} = e^{j\omega_c\tau} \sum_k c_k(t) e^{-j\omega_c\tau_k(t)} = e^{j\omega_c\tau} \cdot f(t). \tag{2.19}$$

If there is no line of sight (NLOS), it is reasonable to assume that all paths have more or less the same attenuation, i.e. all c_k are the same. Another reasonable assumption in this case is that all paths are equally likely to be taken, or in other words the delays $-\omega_c\tau_k$ can be replaced with random variables ϑ_k that are uniformly distributed on $[0, 2\pi)$ [1, 11]; physically this can be imagined as a "ring of scattering objects" around the receiver [9] as sketched in Fig. 2.9(b) (without the red line of sight signal). Finally, assuming that there are infinitely many paths the random variable for the multiplicative fading noise becomes

$$f = \lim_{N\to\infty} \frac{1}{\sqrt{N}} \sum_{k=1}^{N} e^{j\vartheta_k}, \tag{2.20}$$

where the c_k where omitted, since they are assumed to be all equal [1]. The factor $1/\sqrt{N}$ is introduced such that $\mathrm{E}\{|f|^2\} = 1$. It then can be shown that the probability density function of $|f|$ is

$$p(a) = 2ae^{-a^2}, \text{ or } |f| \sim \text{Rayleigh}, \tag{2.21}$$

i.e. the amplitude of f is *Raileigh* distributed [1]. The probability density function of a Rayleigh distributed random variable is shown in Fig. 2.9(a).

LOS case Extending the previous NLOS case, if there is a line of sight (LOS) path (red signal in Fig. 2.9(b)), the statistical model has to be extended by defining the so called Rice factor K. This K factor is the ratio between the power from the LOS path and the average power of the NLOS paths (often also referred to as distributed components) [1]. Hence, by taking K into account (2.20) becomes [1]:

$$f = \sqrt{\frac{K}{K+1}} + \frac{1}{\sqrt{K+1}} \left(\lim_{N \to \infty} \frac{1}{\sqrt{N}} \sum_{n=1}^{N} e^{j\vartheta_k} \right). \tag{2.22}$$

Notice that by letting $K = 0$, that is, no power in the LOS path, (2.22) becomes (2.20) or Rayleigh distributed (as expected). For a factor $K = 5.1$ the probability function is gaussian distributed. Conversely when $K \to \infty$, i.e. no power in the NLOS paths, then $f \to 1$ and the fading disappears. The new amplitude density in this case is:

$$p(a) = 2a(1+K)\exp\left(-K - a^2(K+1)\right) I_0 \left(2a\sqrt{K(1+K)}\right), \tag{2.23}$$

where I_0 the zeroth order modified Bessel function. Random variables with this probability density function are said to have a *Rice* or *Rician* distribution [1].

3 Implementation

3.1 Overview

To illustrate the effects of multipath fading on a wireless transmission link, two machines with an USRP B210 each (hardware) were set up to run a software defined radio (SDR) using the GNU Radio (GR) signal processing toolkit. The latter can also be used as standalone (without hardware) to simulate a channel model. Therefore, simulations of different scenarios affected by multipath fading were elaborated and constructed first using a static FIR model[1], and then with a dynamic statistical model[2]. To present the results a graphical interface was made using the Dear IMGUI framework.

The rest of the chapter is structured as follows. First the tools used in this project are introduced. Then implementations of the transmitter and receiver chains is explained. Subsequently simulations and measurements of fading channels with their respective empirically computed bit error rate (BER) are presented. Finally, the current state of the implementation and issues are discussed.

3.2 Software Stack

3.2.1 GNU Radio

For both the signal processing and the simulations the GNU Radio (GR) toolkit was chosen, as it already had integrations with the USRP hardware drivers. GR is an open-source free software framework that can be used to build signal processing chains and SDRs. GR is composed of two parts: a C++ library with Python bindings to write signal processing code, and GNU Radio Companion (GRC). GRC is a graphical user interface made to more easily construct signal processing chains. Signal processing algorithms are graphically shown as “blocks” that can be chained together with arrows, essentially drawing a diagram called “flow graph”. An example of a flow graph can be seen in Fig. 3.2.

Internally GR works by keeping multiple memory buffers of samples, that are passed as pointers to the signal processing algorithms’ “work functions”. When the signal processing is complete, the output buffer of one block is given to the next block as input according to how they were connected in the flow graph. The structure of a block is shown in the Python listing 3.1. To improve performance GR creates a thread for each work function to parallelize the workload of the concurrently running signal processing blocks. For more details see the GNU Radio Wiki and User Manual in [7].

3.2.2 Dear PyGUI

To construct a graphical interface for a demonstration platform the Dear IMGUI (immediate mode graphical user interface) library was chosen, mainly for its ease of use, wide range of

[1] As discussed in sections 2.4.4 and 2.4.5

[2] See section 2.4.6

```
class myblock(gr.sync_block):
    # there are also other types of blocks such as interpolators
    # (more outputs that inputs), decimators (more inputs than
    # outputs) sync blocks have a 1-to-1 input to output ratio
    def __init__(self, ⟨parameters⟩):
        gr.sync_block.__init__(self, name="My Block",
            # this block as one input port and one output port
            # with samples that are 64 bit complex numbers
            in_sig=[np.complex64], out_sig=[np.complex64]
        )

    def work(self, inputs, outputs):
        # signal processing goes here
        # inputs and outputs are k x n arrays, where each
        # of the k rows is a port that contains n samples
        return ⟨N of inputs that were processed⟩
```

Listing 3.1 A minimal GNU Radio block in Python.

technical capabilites and high refresh rate. Dear PyGUI (DPG) are the Python bindings for the Dear IMGUI library.

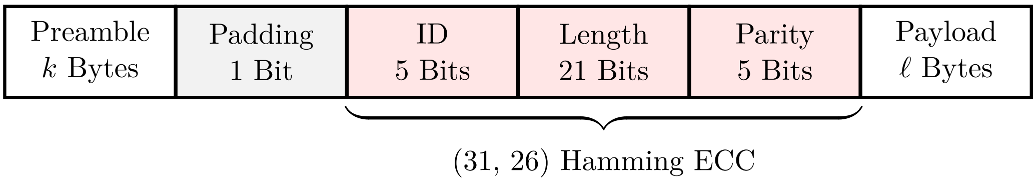
The DPG front-end communicates with the GR flow graphs using the IP/UDP protocol. This decision to separate the project into two parts that communicate over the IP network was made because it is not easy to extend the graphical interface of GRC without interfering with the sophisticated multi-threaded architecture of GR. Furthermore, this allows to have multiple correctly configured flow graphs on disk and to choose which one to run and display on the graphical interface, instead of having a single flow graph whose parameters need to be changed each time. As a side effect, theoretically this setup allows to have one computer running the graphical interface, and another remote machine running just the flow graph. Though the latency caused by the UDP/IP could be substantial.

3.3 Hardware

Dimensions	9.7 × 15.5 × 1.5 cm
Ports	2 TX, 2 RX, Half or Full Duplex
RF frequencies	70 MHz to 6 GHz
Bandwidth	200 kHz — 56 MHz
External reference input	10 MHz

Table 3.1 USRP B210 specifications [17].

As already mentioned, for the receiver and transmitter in the SDR setup two USRP B210 devices from Ettus Research were used. Some technical specifications of the device are shown



Preamble k Bytes	Padding 1 Bit	ID 5 Bits	Length 21 Bits	Parity 5 Bits	Payload ℓ Bytes

Figure 3.1 Structure of framed data packets used in the implementation.

in Tab. 3.1. GR provides off the shelf blocks that interface with the official API provided from Ettus Research.

3.4 Transmitter chain

3.4.1 Data frame

To compute the empirical bit error rate (BER) of the setup, the data has to be framed by the transmitter and the bitstream needs to be synchronized on the receiver side. The structure of a data packet used in the implementation is shown in Fig. 3.1. A frame begins with a user specified k-byte preamble, that in the current implementation serves as synchronization pattern. Another use case for the preamble sequence could be to introduce channel estimation pilot symbols (this will be discussed more in section 4.2.3). Following the preamble are 4 bytes encoded using a (31, 26) Hamming code (plus 1 padding bit), that contain metadata about the packet, namely payload ID and payload length. Because the payload length in bytes is encoded in 21 bits, the theoretical maximum payload size is 2 MiB, which together with 32 possible unique IDs gives a maximum data transfer with unique frame headers of 64 MiB. These constraints are a result of decisions made to keep the implementation simple.

3.4.2 Modulation

GR provides a constellation modulator block, that already implements several standard constellations (QPSK and 16-ary QAM being of interest for us). The block also already integrates a root raised cosine filter, whose phase bandwidth (roll-off factor) can be given as parameter; in all flow graphs the roll off factor is $\alpha = 0.35$.

3.5 Receiver chain

3.5.1 Envelope detector

What is here referred to as envelope detector has the purpose of synchronizing the symbols and equalizing the input signal amplitude. This is accomplished in GRC using two blocks: a polyphase clock sync and a constant modulus adaptive (CMA) filter equalizer. The input signal for the envelope detector has 4 samples per symbol, while the output has only one sample per symbol. The choice of the CMA equalizer later turned out to be a mistake in the QAM flow graphs, as it only works for envelopes with a constant amplitude. In the latest version least mean square decision-directed (LMS-DD) equalizers have been used.

3.5.2 Frame synchronization and phase correction

Once the envelope's clock is synchronized in the processing chain the data stream has one sample per symbol. At this point it is necessary to find where each data frame starts or end in order to correctly decode their payloads. For such purpose a special sequence called *access code* is put in front of each frame. Access codes are sequences of samples that are carefully constructed to have an autocorrelation with a high peak at zero, and that rapidly decreases for increasing shifts. In other words, the autocorrelation of an access code high only when the sequence is perfectly aligned. Thus, by cross correlating an envelope signal $r(t)$, that periodically contains an access code $a(t)$ with the access code itself, and looking for peaks in the result, it is possible to determine where each frame begin. Furthermore, by analyzing the values of the peaks it is possible to extract information about the phase and frequency offsets.

To understand how correlation peaks allow for fine phase correction, recall that the cross correlation (denoted here by $\star$) of two complex valued signals is

$$R_{ra} = (r \star a)(t) = \int_{\mathbb{R}} r(\tau) a^*(\tau - t)\, d\tau = r(t) * a^*(-t), \tag{3.1}$$

which is equivalent to a convolution, with the left term being time-reversed complex conjugated [10]. This last property is especially useful because it makes possible to implement cross correlation using FIR filters. Some interesting properties of the cross correlation are that correlation with itself (autocorrelation) at $t = 0$ is

$$R_{aa} = (a \star a)(0) = \int_{\mathbb{R}} a(\tau) a^*(\tau - 0)\, d\tau = \int_{\mathbb{R}} |a(\tau)|^2\, d\tau \in \mathbb{R}, \tag{3.2}$$

which is a real number. And more importantly the correlation with an out of phase copy $a'(t) = a(t)e^{j\varphi}$ at $t = 0$ is

$$(a' \star a)(0) = \int_{\mathbb{R}} a(\tau) e^{j\varphi} a^*(\tau)\, d\tau = R_{aa} e^{j\varphi}. \tag{3.3}$$

The relevant observation to make in (3.3) is that since R_{aa} is a real number, the phase of the cross correlation at $t = 0$ is the phase of $a'(t)$. This fact can be exploited to implement fine phase correction for the received envelope in relatively few steps as follows:

1. Compute the cross correlation R_{ra} of the envelope $r(t)$ with the access code $a(t)$,
2. Find the maximum value of $\hat{R}_{ra} = \max R_{ra}(t)$ (correlation peak),
3. Extract the phase offset $\varphi = \arg \hat{R}_{ra}$,
4. Remove the phase offset in the envelope by multiplying it with the complex conjugate of the offset, that is $\hat{r}(t) = r(t)e^{-j\varphi}$.

Implementing fine phase and frequency correction

To implement in GR what was discussed above the two blocks shown in Fig. 3.2 were used: a correlator estimator block, and a custom block. The former essentially implements the first 3 of the steps discussed at the end of the previous subsection. The correlator estimator block

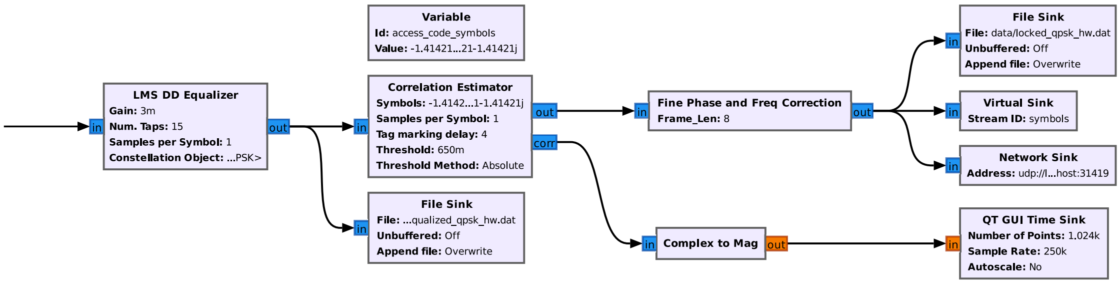


Figure 3.2 Part of the GNU Radio flow graph for the QPSK modulated link (with hardware). The shown blocks are used to equalize and lock the envelope.

is given a sequence of samples, and when the cross correlation between them and the input stream is higher than a certain threshold (90% of the amplitude of a perfect autocorrelation), it produces a "tag" in the output stream, that contains the phase estimate.

Tags are GR's way of working with metadata that is attached to a sample. Internally tags are just polymorphic data structures containing a number indicating the absolute offset (in samples), and a pair of arbitrary values called "key" and "value". Tags are passed on from one block to the next like sample streams (unless the block specifies to do otherwise).

Thus, the tagged stream is processed with a custom block, of which a simplified version of its work function shown in listing 3.2. The custom block also implements fine frequency correction (shown in listing 3.3) by linearly interpolating the phase estimates between each pair of tags (called chunk). This can be rather trivially be formulated for a chunk of N samples as the

$$k\text{-th chunk digital frequency} \quad \Omega_k = (\varphi_{k+1} - \varphi_k)/N, \text{ and the} \tag{3.4a}$$

$$k\text{-th chunk phase estimate} \quad \Phi(n) = \varphi_k - \omega_k n/N. \tag{3.4b}$$

Performance of the implementation

The phase and frequency correction block was implemented with the design goal of being able to correct a maximum frequency offset of $\hat{\epsilon} = 0.1\%$ under ideal conditions, which is sufficient to take into account small Doppler shifts at walking speed ($v = 2\,\mathrm{m\,s^{-1}}$) with carrier at $f_c = 2.4$ GHz. The USRP B210 devices have an internal clock frequency accuracy of $\epsilon = 1$ ppm $= 10^{-6}$, which results in a total frequency offset of

$$\Delta f = f_c \left(\frac{v}{c_0} + \epsilon \right) = 2.4\,\mathrm{GHz} \left(\frac{2\,\mathrm{m\,s^{-1}}}{3 \times 10^8\,\mathrm{m\,s^{-1}}} + 10^{-6} \right) = 2416\,\mathrm{Hz}. \tag{3.5}$$

Because the frequency estimate is linearly interpolated, the phase error may not exceed π (half rotation) during one data frame (chunk). These constraints imply that for frames of N' symbols of duration T each, using κ samples per symbol the relation

$$2\pi \Delta f N' T \kappa \leq \pi \implies T = 1/f_s \leq \frac{1}{2\Delta f N' \kappa}, \iff N' \leq \frac{1}{2\Delta f T \kappa}, \tag{3.6}$$

must hold. By further setting $\kappa = 4$ and $N' = 32$ we obtain a minimum sampling frequency of approximately 618.5 kHz, or conversely by letting $f_s = 1\,\mathrm{MHz}$ we have a maximum frame length of $N' = 51$ symbols. In other words, roughly every 50 symbols the system must send an access code sequence. This result is rather unfortunate as it means that more processing power than expected is required.

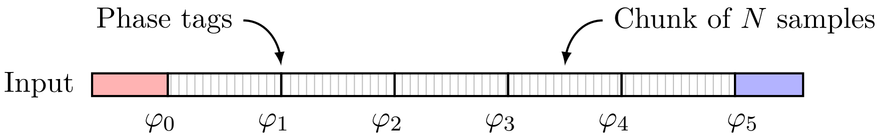


Figure 3.3 Graphical representation of the input samples for the work function of the fine phase and frequency correction block (shown in listing 3.2). Roughly every N samples there is a tag containing the information of the phase error (computed using the cross correlation peak). The white ‘chunks’ of samples can be corrected using their respective left and right tag values. The samples in the red chunk need phase information from the previous block processing. The samples in the blue chunk need a phase information from the future, which is not attainable. Thus for the blue chunk the frequency estimate of the previous chunk is used.

```
def work(self, inputs, outputs):
    # alias for inputs of the first port
    inp = inputs[0]
    # read phase tags from stream
    is_phase = lambda tag: pmt.to_python(tag.key) == "phase_est"
    tags = filter(is_phase, self.get_tags_in_window(0, 0, len(inp)))
    # create a list of pairs $((\varphi_0, \varphi_1), (\varphi_1, \varphi_2), \ldots, (\varphi_{k-1}, \varphi_k))$
    pairs = zip(tags, tags[1:])
    # compute phase correction between each pair of tags
    chunks = [self.block_phase(start, end) for (start, end) in pairs]
    # flatten array to get $\Phi(n)$ and compute the correction
    phases = np.concatenate(chunks)
    correction = np.exp(-1j * phases)
    # write to the first output port
    left = tags[0].offset - self.nitems_written(0)
    right = tags[-1].offset - self.nitems_written(0)
    outputs[0][left:right] = inp * correction
    # return how many samples were processed
    return len(outputs[0])
```

Listing 3.2 Simplified work function of fine phase correction block that corrects only samples ‘in the middle’. The version that is actually used does handle edge cases that have been removed here for readability. See also Fig. 3.3 for a graphical representation of the inputs and listing 3.3 for the definition of the `block_phase` function.

```python
    def block_phase(self, start, end):
        # compute number of samples between tags
        nsamples = end.offset - start.offset
        # unpack pmt values into start and end phase
        sphase = pmt.to_python(start.value)
        ephase = pmt.to_python(end.value)
        # compute frequency offset between start and end
        phasediff = (ephase - sphase)
        freq = phasediff / nsamples
        if phsediff > np.pi:
            phasediff -= np.pi
        elif phasediff < -np.pi:
            phasediff += np.pi
        # compute chunk values
        return sphase * np.ones(nsamples) + freq * np.arange(0, nsamples)
```

Listing 3.3 Block phase function referenced in listing 3.2.

3.5.3 GUI implementation

The GUI is implemented with the Dear PyGUI library as introduced in section 3.2.2. In Fig. 3.4 the graphical interface of it is shown. Four different constellation plots are illustrated: the channel itself, after the polyphase clock synchronization, after the equalizer and the locked constellation. The GUI shows the BER of the constellation and a time domain plot. The interface also contains a block diagram.

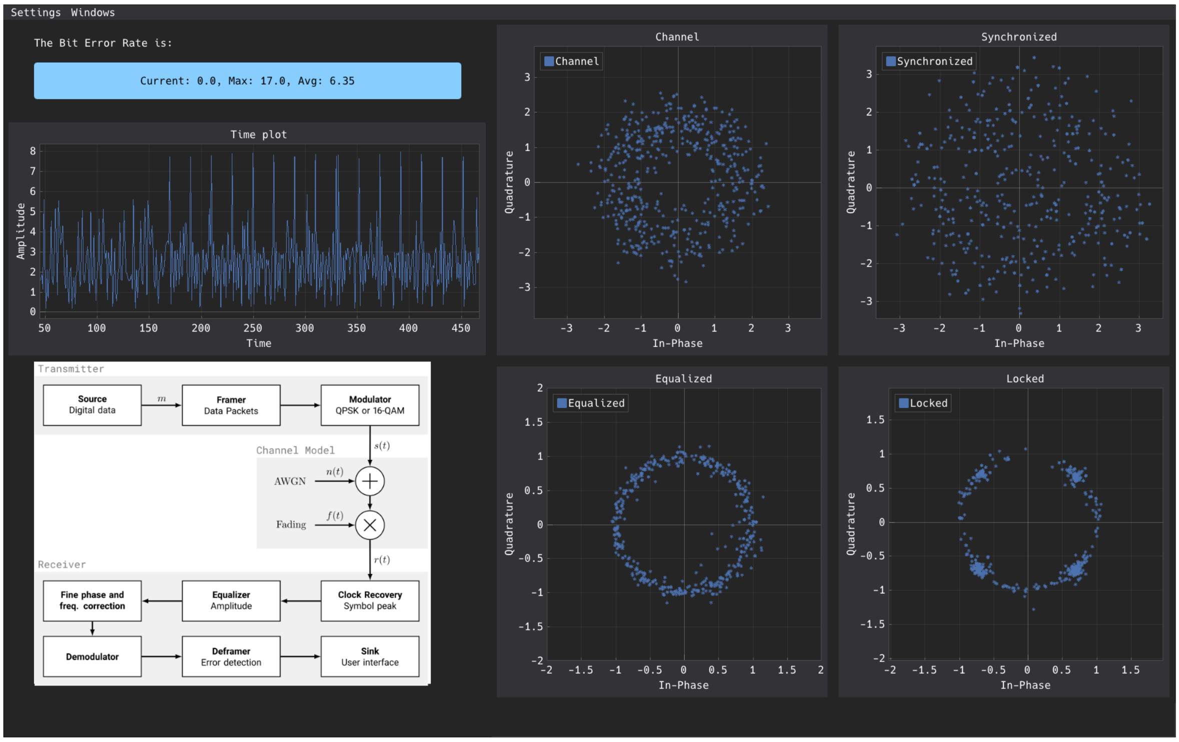
3.6 Channel simulations

In order to study the effects of multipath fading, a series of simulations have been made under different conditions. To simulate a channel affected by multipath fading two blocks from the GR library, and a third custom block were used. The channel model can simulate AWGN, a frequency offset and either a Rayleigh (NLOS) oder Rice (LOS) fading.

3.6.1 Fading with discrete time model

To implement and illustrate a static fading effect corresponding to section 2.4.4, a custom block was created. The work function of the block is shown in listing 3.4. This block is based on a FIR filter. It can be configured to simulate a with direct path (LOS) or without (NLOS).

In this block it is possible to simulate any number of paths with different attenuations. A special case is show in the first and second colum of Fig. 3.6, where the delay is a multiple of the symbol time. Another example with more taps is shown in the third colum. These simulation values are however not realistic, because the static model is too simple. It does not account for other effects, such as changes in the environment. The block additionally implemented the method described in section 2.4.5 to allow non-integer delays. Here the sinc



The Bit Error Rate is:

Current: 0.0, Max: 17.0, Avg: 6.35

Figure 3.4 Screenshot of the graphical interface of receiver built using the DearPyGUI library.

pulse was limited using a simple boxcar function, the filter could be improved by using for example a Hann window.

3.6.2 Fading with statistical model

In order to represent the effect of multipath fading not only statically, a second statical model fro GR was used. The block named Frequency Selective Fading Model is implemented using the algorithm from [3], with the help of the sum-of sinusoid principle (SOS). The algorithm in this block is implemented with the aim that only a small number of sinusoids are needed to simulate each ray. For the simulations shown, the recommended value of 8 has been chosen.

In the block it is possible to choose between Rayleigh or Rician for the statistical modeling. When the Rician model is chosen, a realistic value for the K factor (between zero and ten [8]) needs to be given. As mentioned in section 2.4.6, if $K = 0$ the distribution is the same as in the Rayleigh model. The delays of the power profile are specified in samples. Those vectors depend on the environment. The magnitudes of the pulses are given in their linear values. In practice the average path gain of a fading path is in the range from $-20\,\text{dB}$ to $0\,\text{dB}$. To add movement, some Doppler shift can be introduced, but this frequency offset needs to be normalized with the sampling rate.

Issues

It is difficult to check, whether the noise generated from the statistical model is correct. Especially when the Doppler effect is included. Then the simulation is difficult to recreate, when the amplitude and phase parameter are not in a special state, in which the amplitude

```
def work(self, input_items, output_items):
    inp = input_items[0]
    oup = output_items[0]
    # find the length of the highest order filter
    max_order = 2 * np.floor(np.max(self.delays)) + 1
    max_samples = np.arange(0, max_order +1)
    max_len = len(max_samples)
    # total impulse response (of all taps)
    tot_h = np.zeros(int(max_len))
    # compute for each tap
    for (a, d) in zip(self.amplitudes, self.delays):
        # compute fir coefficients
        order = 2 * np.floor(d) + 1
        samples = np.arange(0, order + 1)
        # compute impulse response
        h = a * np.sinc(samples - d)
        # correct length
        h = np.concatenate([h, np.zeros(max_len - len(h))])
        # add to other filters
        tot_h += h
    # add a LOS path if necessary
    tot_h[0] = self.los
    # compute output
    y = np.convolve(inp, tot_h)
    # add values from previous block processing
    y += np.concatenate([self.temp, np.zeros(len(y) - len(self.temp))])
    # write to output
    oup[:] = y[:len(inp)]
    # save rest for next block processing
    self.temp = y[len(inp):]
    return len(oup)
```

Listing 3.4 Implementation of a static fractional delay FIR filter.

and the phase shift can be seen exactly. To have a possibility to verify the plot, mainly whether the movement of the signal could be correct, a Matlab model with the same values as in the GR simulation was used. Using this method, the model turned out to be correct.

Example with realistic values

Excess tap delay	Relative power	Excess tap delay	Relative power
0 ns	$-1.0\,\text{dB} \approx 0.7943$	1.6 µs	$-3.0\,\text{dB} \approx 0.5011$
50 ns	$-1.0\,\text{dB} \approx 0.7943$	2.3 µs	$-5.0\,\text{dB} \approx 0.3162$
120 ns	$-1.0\,\text{dB} \approx 0.7943$	5.0 µs	$-7.0\,\text{dB} \approx 0.1995$
200 ns	$0.0\,\text{dB} = 1.0000$		
230 ns	$0.0\,\text{dB} = 1.0000$		
500 ns	$0.0\,\text{dB} = 1.0000$		

Table 3.2 Extended Typical Urban model (ETU) ETSI Standard PDP values for multipath fading propagation conditions [16].

In order to obtain a realistic simulation, the values for multipath fading propagation conditions of an Extended Typical Urban (ETU) model, from the ETSI (European Telecommunication Standards Institute) were used [16]. The values shown in Tab. 3.2. For these values the maximum Doppler frequency possibilities are predefined. In the following examples in Fig. 3.7 either 5 Hz or 70 Hz were used, opposed to the are values calculated in (3.5) for a walking speed of $2\,\text{m}\,\text{s}^{-1}$, where the Doppler frequency is 16 Hz. Those predefined values correspond to a speed of

$$v_1 = \frac{\Delta f}{f_c} \cdot c_0 = \frac{5\,\text{Hz}}{2.4\,\text{GHz}} \cdot 3 \times 10^8\,\text{m}\,\text{s}^{-1} = 0.625\,\text{m}\,\text{s}^{-1}, \text{ and} \tag{3.7}$$

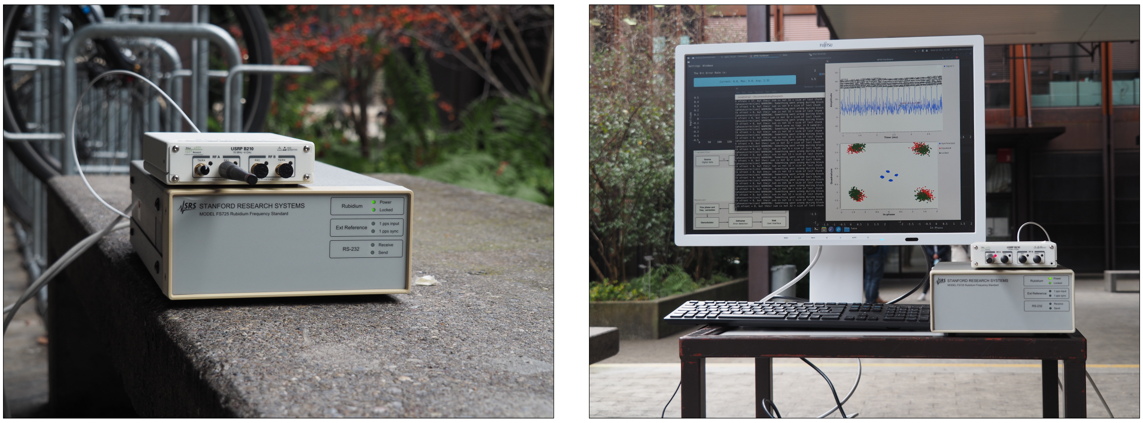
$$v_2 = \frac{\Delta f}{f_c} \cdot c_0 = \frac{70\,\text{Hz}}{2.4\,\text{GHz}} \cdot 3 \times 10^8\,\text{m}\,\text{s}^{-1} = 8.75\,\text{m}\,\text{s}^{-1}. \tag{3.8}$$

The numbers of taps used in this case are the number of given values.

3.6.3 Measurements

To demonstrate the fading effect, two SDRs were used. For that multiple different measurements were made. For example one in an indoor environment, the Lab. Another in an outdoor environment: The set up is shown in Fig. 3.5. The results of those measurements are illustrated in Fig. 3.9. Because of the USB cables the distance between the two SDRs was only about 2 to 3 m. The signals were all sent with a gain value of 0.4. The phase and amplitude changes could be seen well. Especially when the transmitter or the receiver were moved, the change of them got faster.

The BER, which will be described in detail in the next section, on average was 2.37 for the outdoor environment and about 2.67 for indoors. It makes sense that the fading effect occurs more in an indoor environment, because there were more surfaces for reflections.



(a) Reviser SDR in the outdoor environment set up.

(b) Transmitter SDR in the outdoor environment measurement set up.

Figure 3.5 Outdoor measurement set up.

3.6.4 Empirical BER

To find out how accurate the simulations are compared with a simulation of the fading effect and measurements, the bit error rate of the system is calculated. This is done with the help of a user specified 2-byte test payload followed by 2 randomly generated bytes. As seen in the simplified listing 3.5, every bit is compared with the test vector after the demodulation stage. Because of the fact that the test vector has some random bits at the end, the bit error rate has always an average value of 32, even if the test vector perfectly matches. To only focus on the true BER of the signal, this value is subtracted. The random byte at the ends were introduced to help the feedback loops in the synchronization and equalization stages, which would otherwise would drift and lose their operating point. The test vector used is `[0x12, 0x48]`, because these binary numbers are easy to recognize in a continuous bit stream. For generating the bit error rate a bit stream with a specific length is compared with the test vector. To make it simpler and to avoid mistakes, additionally an average BER over the last 200 test vectors is computed.

3.7 Issues in the current implementation

3.7.1 Non modulated access codes

Currently, as described in section 3.4.1, the access codes are put as bytes in front of the frame in the k-byte preamble. For this to work, the access code bytes must still have a good autocorrelation function after being modulated into symbols using the chosen modulation scheme. This works well with QPSK, because the constellation is quite simple and the length of the sequence is only halved after the modulation (since QPKS has 2 bits per symbol). Thus, in the QPSK flow graph the longest known Barker sequence `0x1f35` (13 bits, left padded with zeros) is sufficient (≈ 7 symbols).

With QAM however, the complexity of the constellation and the higher number of bits per symbol makes it increasingly difficult to find binary sequences that retain a good autocorrelation after being modulated. A better solution for example would be to use a *constant*

```
    def work(self, input_items, output_items):
        # input is a list of blocks of k-bytes
        inp = input_items[0]
        # for each block
        for i in inp:
            i = np.array(i, dtype=np.uint8)
            # XOR to compute the difference
            v = np.array(self.vgl, dtype=np.uint8) ^ i
            # compute how many bits differ
            ber = sum(np.unpackbits(v))
            # save BER value
            self.ber_samples.appendleft(ber)
        # compute statistics and send to GUI
        ber_max, ber_min, ber_avg = self.ber_stats()
        self.send(self.encode([ber_max, ber_min, ber_avg]))
        return len(inp)
```

Listing 3.5 Custom block to compute the empirical BER.

amplitude zero autocorrelation waveform (CAZAC) of length N, which is computed with

$$u_k = \exp\left(j\frac{M\pi K}{N}\right) \quad \text{where } K = \begin{cases} k^2 & \text{when } N \text{ is even} \\ k(k+1) & \text{when } N \text{ is odd,} \end{cases} \tag{3.9}$$

and M is relatively prime to N [6]. CAZAC waveforms are ideal because they have a Dirac delta as autocorrelation [6], i.e. $R_{uu}(\tau) = \delta(\tau)$. Unfortunately, since these complex values are not on any constellation point they break some assumptions of the polyphase clock sync and the LMD DD equalizer (but not CMA, which unfortunately cannot be used for QAM). Thus, to use CAZAC waveforms, the transmitter needs to put them in front of the modulated symbols (for example using a correctly parametrized Stream MUX block in GR), and the receiver would need to synchronize with the sequence before the clock recovery or equalization. The latter is especially problematic because then it is no longer possible to identify the peak by comparing the autocorrelation value to a fixed threshold as done in section 3.5.2. Because of the aforementioned reasons, in its current state the QAM flow graph is unable to lock and decode any signals.

3.7.2 Single threaded GUI application

The current GUI prototype built with DearPyGUI has some issues, the most critical begin that it is a single-threaded program. The interprocess communications (with GR's flow graphs) should be on a separate thread from the graphics, what is currently not the case. The problem is not noticeable as long as the flow graphs in the background keep sending data, but as soon as the UDP/IP data stream stops, the timeout of the socket interface causes the interface to run at less than 20 frames per second.

3.7.3 Clock synchronization issues

Unfortunately the two SDR need an external clock generator. For that a Rubidium Frequency standard device (Model FS725) is used with the clock frequency of 10 MHz. Two of them are used to make them more movable and independent. Those clock generators where needed, because the synchronization does not work as planed in 3.5.2. Without those only the amplitudes could be seen in the plots.

3.8 Produced constellation plots

In the next few pages the constellation plots from the simulation results and the hardware measurements are shown.

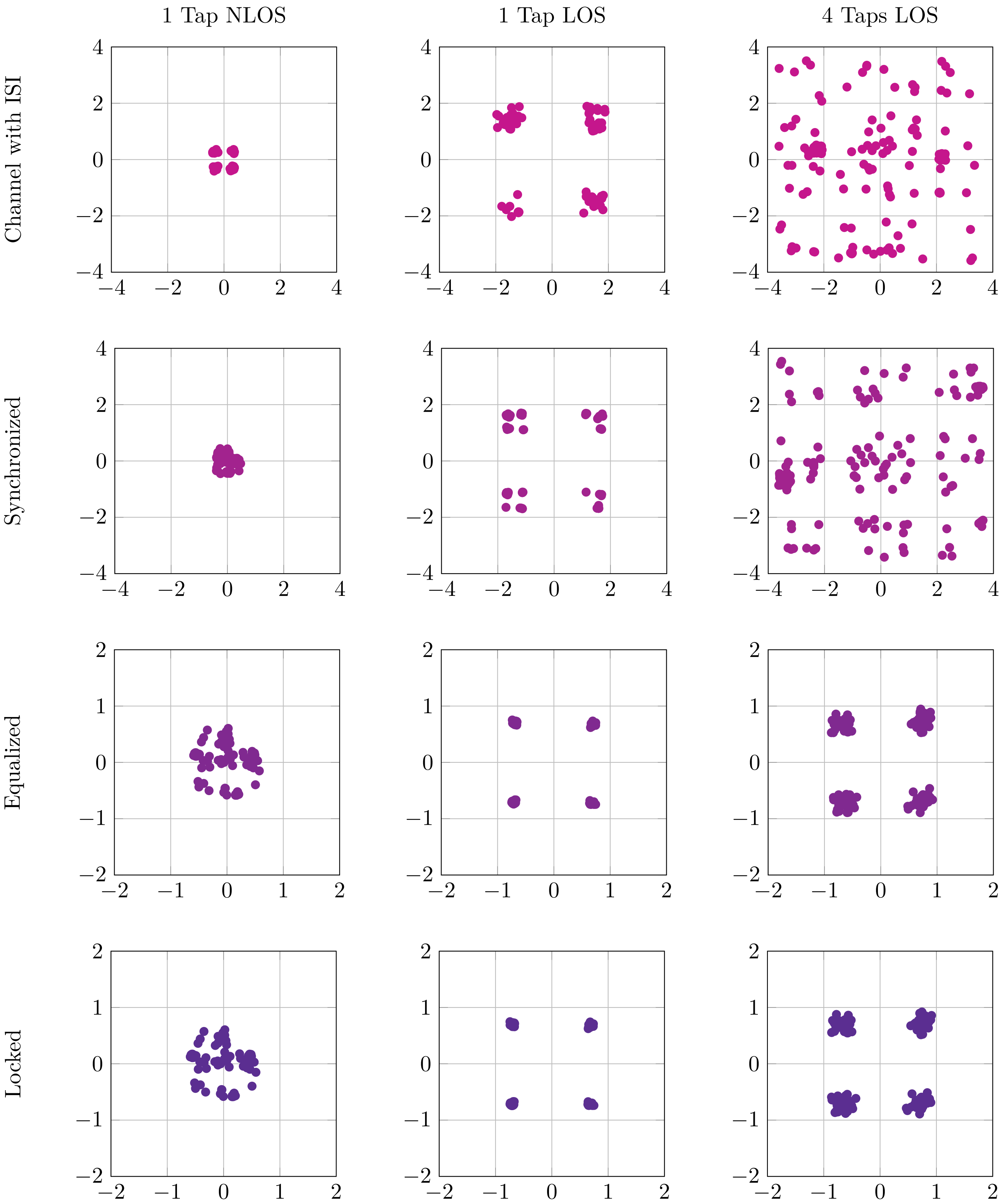


Figure 3.6 Simulations of a static fading channel models with different tap values. The samples were generated using the custom block discussed in section 3.6.1. For the 1 tap model the fading tap was $0.2\delta(n-0.25)$, and for the 4 tap model uses $0.2\delta(n-0.25)+0.08\delta(n-3.25)+0.5\delta(n-4)+0.4\delta(n-6.3)$. In both cases the delays are given in samples.

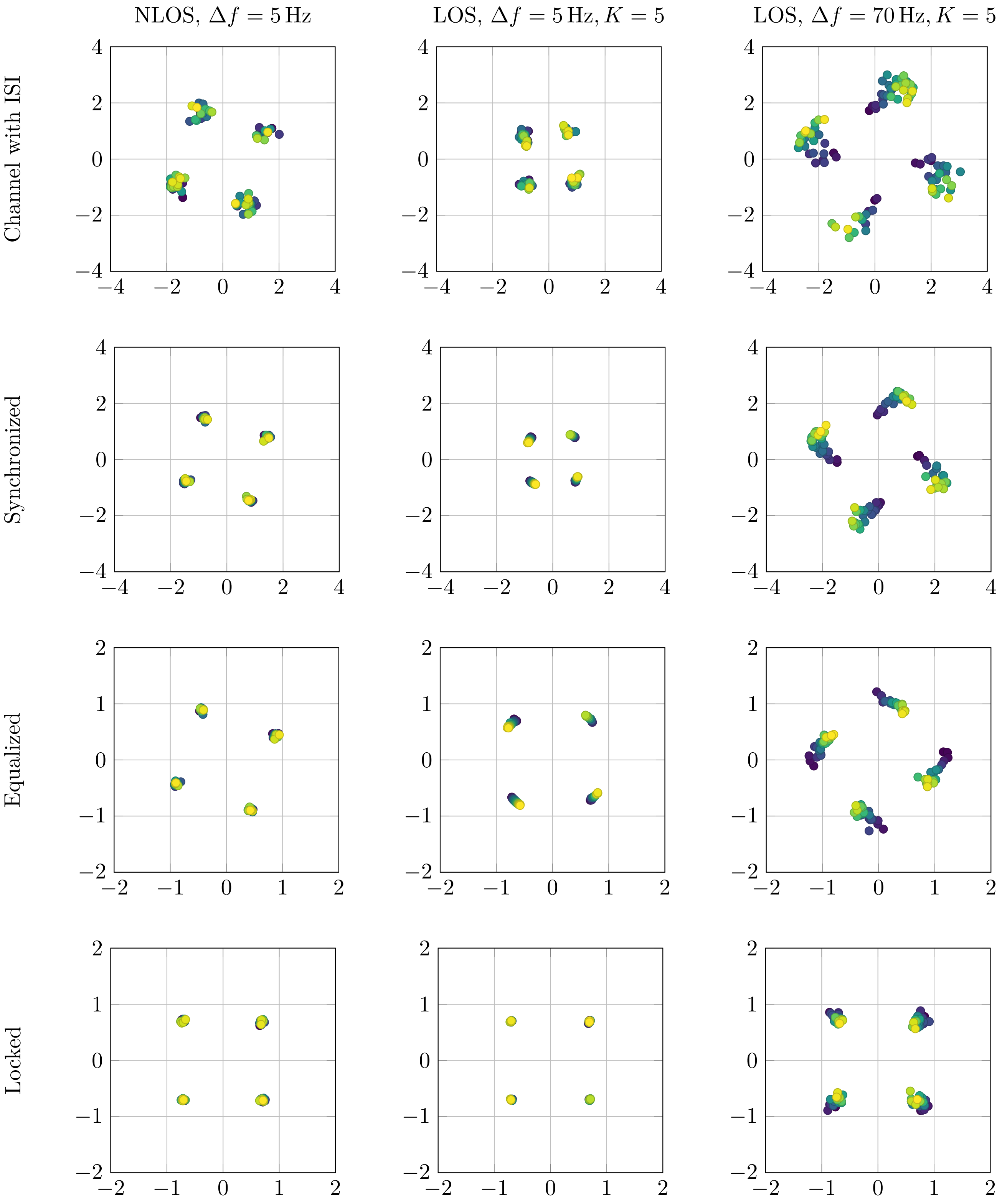


Figure 3.7 Simulations with QPSK modulation and a dynamic fading channel model that uses PDP values of the Extended Typical Urban model (ETU) from the ETSI standard normative Annex B.2 in [16]. The PDP values from the standard are reported in Tab. 3.2. The color gradient represents progression in time: yellow samples are more recent than the blue samples.

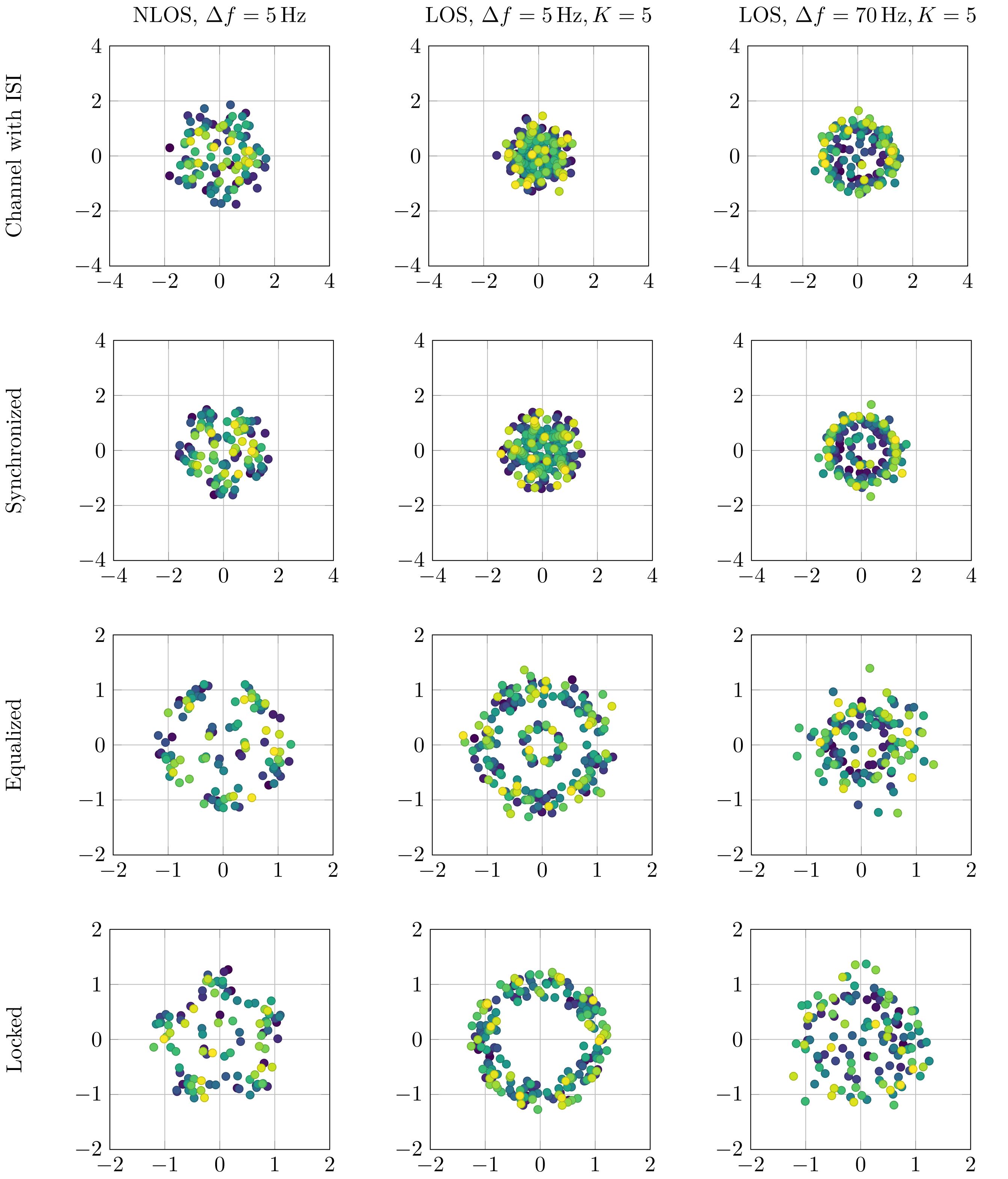


Figure 3.8 Simulation using the same channel parameters as in Fig. 3.7, but with a 16-ary QAM modulation scheme. Unfortunately, because of the issue discussed in section 3.7.1 the receiver cannot lock and decode the envelope.

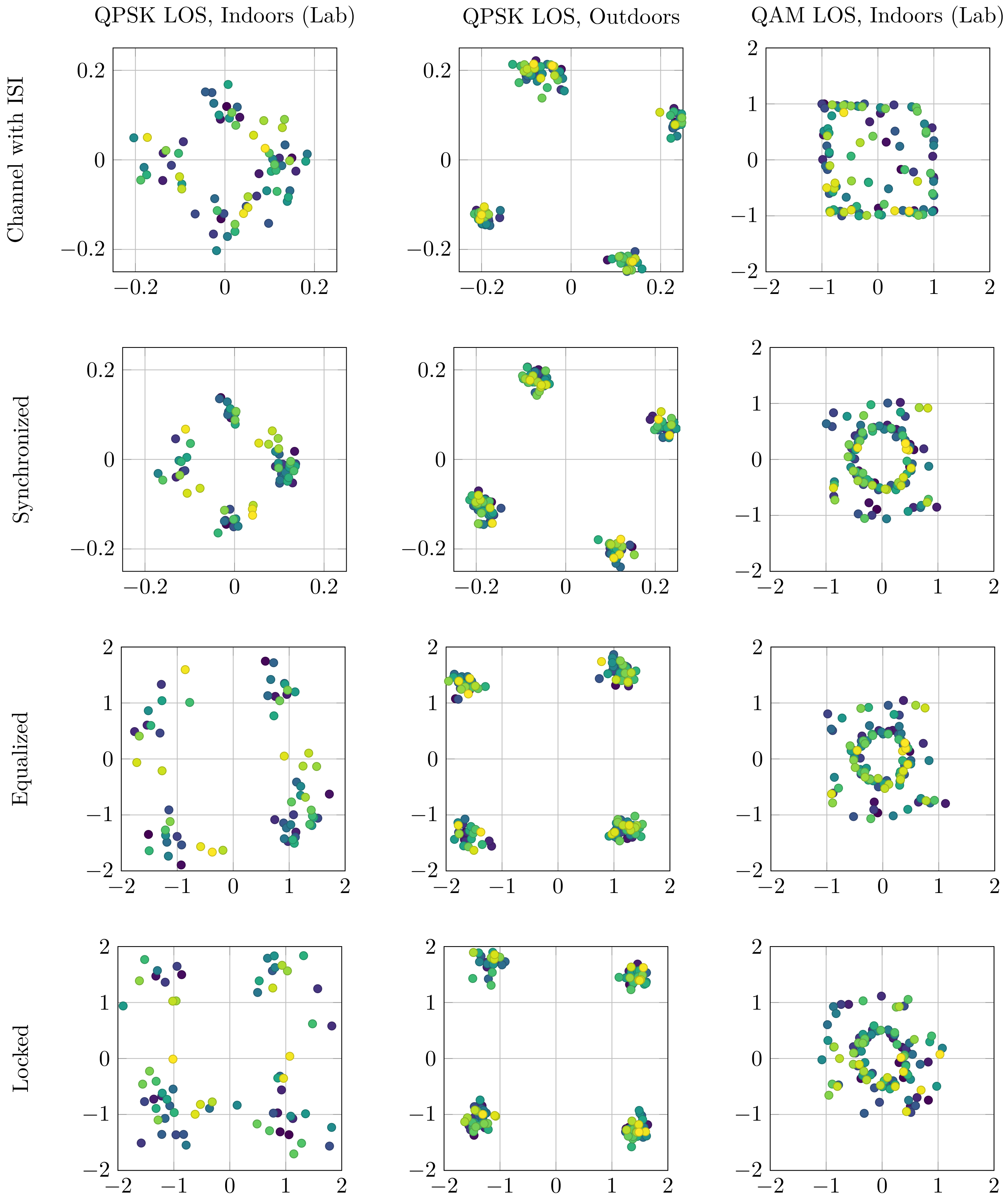


Figure 3.9 Constellation plots from the different measurements with the two SDRs.

4 Conclusions

4.1 Results

The goal to build a functional demonstrator has been only partially achieved, unfortunately not all of the originally planned features could be implemented. A stable wireless link using QPSK modulation that computes the BER was developed. Because of the issue discussed in section 3.7.1, the QAM variant cannot compute the empirical BER.

For both modulation schemes samples from multiple different conditions were collected and analyzed, albeit some assessments could only be conducted when QPSK was used.

4.2 Future Work

4.2.1 Improve BER measurements and simulations

A missing feature in this work is an automated collection of the BER data, which would allow to more easily observe and measure the influence of each parameter in the fading channel model.

4.2.2 Improvements in the GUI front-end

In addition to fixing the issue discussed in section 3.7.2, a very important feature that is currently missing is the ability to change the fading parameters in real time from within the GUI. Dear PyGUI offers many graphical elements that could be used to control the parameters, however a new GR block would need to be created to propagate the updated values into the flow graph.

4.2.3 Channel parameters estimation with PSAM

An interesting continuation of this work could be to regularly interpolate some so called pilot symbols in the modulated data stream. In short, the pilot symbol assisted modulation (PSAM) technique consists of periodically inserting informationless (known) symbols in the data stream, which can then be used to estimate the fading parameters of the communication channel. More details are presented in [20] (and its references) from which the illustrations in Fig. 4.1 were taken.

4.3 Acknowledgments

We would like to thank everyone who took the time to help us. Nicola Ramagnano for his explanations and with the GNU Radio tool, Michel Nyffenegger for his comments and ideas, Marcel Kluser, for all of the equipment he provided to us. Our friends who supported us in different ways, especially Manuel Kritzer and Manuel Spuhler for the correction reading.

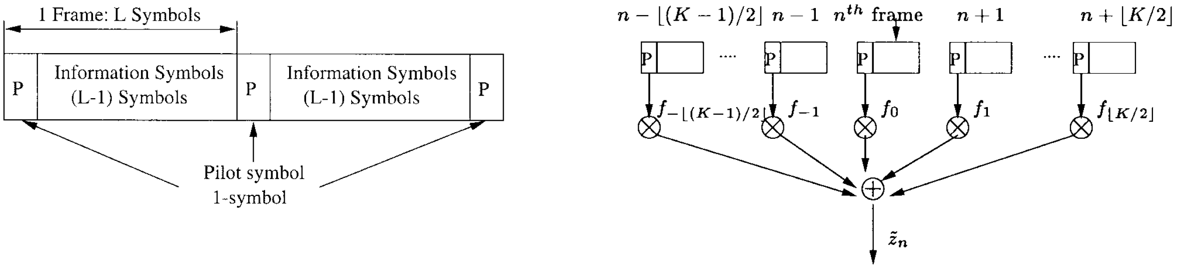


Figure 4.1 Illustration of the pilot symbols assisted modulation (PSAM) frame format (left), and PSAM fading interpolation method (right). The PSAM technique allows to compute the fading (denoted in the figure with $\tilde{z}_n$) by interpolating measurements of informationless symbols (pilot symbols) over multiple frames. Both figures were taken from [20], which presents an analytical method to compute the BER from the PSAM and multilevel quadrature amplitude modulation (M-QAM) parameters.

Bibliography

[1] Hoher Peter Adam. *Grundlagen der digitalen Informationsübertragung*. de. 2nd ed. Springer Vieweg, June 2013. ISBN: 978-3-8348-1784-6. DOI: `10.1007/978-3-8348-2214-7`.

[2] Macmillan Alex, Marina Mahesh K., and Triana Jhair Tocancipa. "Slow Frequency Hopping for Mitigating Tidal Fading on Rural Long Distance Over-Water Wireless Links." In: *2010 INFOCOM IEEE Conference on Computer Communications Workshops*. 2010, pp. 1–5. DOI: `10.1109/INFCOMW.2010.5466612`.

[3] Alimohammad Amirhossein et al. "Compact Rayleigh and Rician fading simulator based on random walk processes." In: *IET communications* 3.8 (2009), pp. 1333–1342.

[4] Frederiksen Flemming Bjerge and Prasad Ramjee. "An overview of OFDM and related techniques towards development of future wireless multimedia communications." In: *Proceedings RAWCON 2002. 2002 IEEE radio and wireless conference (Cat. No. 02EX573)*. IEEE. 2002, pp. 19–22.

[5] Antonescu Bogdan, Moayyed Miead Tehrani, and Basagni Stefano. "mmWave channel propagation modeling for V2X communication systems." In: *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. 2017, pp. 1–6. DOI: `10.1109/PIMRC.2017.8292718`.

[6] Chu D. "Polyphase codes with good periodic correlation properties (Corresp.)" In: *IEEE Transactions on Information Theory* 18.4 (1972), pp. 531–532. DOI: `10.1109/TIT.1972.1054840`.

[7] GNU Radio developers. *GNU Radio Wiki*. 2021. URL: `https://wiki.gnuradio.org`.

[8] *Fading Channels*. The MathWorks, Inc. Dec. 2021. URL: `https://www.mathworks.com/help/comm/ug/fading-channels.html`.

[9] Prof. Dr. Messier G. *Wireless Communications: Small Scale Fading*. 2018. URL: `https://www.youtube.com/watch?v=4ibjrRzvJ5E&list=PL7sWxFnBVJLXsvsLzMXT2Fk4ZPD7BJTBX&index=3`.

[10] Robert Gallager. *Course material for 6.450 Principles of Digital Communications I*. Massachusetts Institute of Technology MIT. 2006. URL: `https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-450-principles-of-digital-communications-i-fall-2006`.

[11] Prof. Dr. Mathis H. and Prof. Dr. Lang H.D. *Wireless Communication*. Eastern Switzerland University of Applied Sciences, Sept. 2021.

[12] Hsu H.P. *Schaum's Outline of Analog and Digital Communications*. Schaum's outline series. McGraw-Hill Education, 2003. ISBN: 9780071402286.

[13] Frank R. Kschischang. "The Hilbert Transform." In: (Oct. 2006). URL: `http://web.eecs.utk.edu/~mjr/ECE342/hilbert.pdf`.

[14] Subramanian Lakshminarayan et al. "Rethinking wireless for the developing world." In: *IRVINE IS BURNING* (2006), p. 43.

[15] Maddocks M.C.D. and Newell C. “An introduction to digital modulation and OFDM techniques.” In: *BBC Research Department* 10 (1993).

[16] ETSI 3rd Generation Partnership Project. *ETSI TS 136 104 V11.4.0.* Technical Specification. European Telecommunication Standards Institute. URL: `https://www.etsi.org/deliver/etsi_ts/136100_136199/136104/11.04.00_60/ts_136104v110400p.pdf`.

[17] Ettus Research. *USRP^{TM} B200/B200 Bus Series Specification Sheet.* 2019. URL: `https://www.ettus.com/wp-content/uploads/2019/01/b200-b210_spec_sheet.pdf`.

[18] Flickenger Rob. *Wireless Networking in the Developing World: A practical guide to planning and building low-cost telecommunications infrastructure.* Hacker Friendly LLC, Seattle, WA, US, 2007.

[19] Kneubühler T. *Nachrichtentechnik 1 + 2.* Eastern Switzerland University of Applied Sciences, Sept. 2020.

[20] Xiaoyi Tang, M.-S. Alouini, and A.J. Goldsmith. “Effect of channel estimation error on M-QAM BER performance in Rayleigh fading.” In: *IEEE Transactions on Communications* 47.12 (1999), pp. 1856–1864. DOI: `10.1109/26.809706`.

[21] Välimäki Vesa. “Discrete-Time Modeling of Acoustic TubesUsing Fractional Delay Filters.” Helsinki University of Technology, Laboratory of Acoustics and Audio Signal Processing, Dec. 1995. ISBN: 951-22-2880-7.

A Appendix

A.1 Certificaton of authorship

We hereby certify that this thesis has been composed by us and is based on our own work, unless stated otherwise. No other person's work has been used without due acknowledgement in this thesis. All references and verbatim extracts have been quoted, and all sources of information, including graphs and data sets, have been specifically acknowledged.

Place	**Date**
Rapperswil	23. December 2021

Signatures

Naoki Sean Pross

Sara Cinzia Halter

A.2 Attached documents

- Task Description
- Project Plan

OST
Ostschweizer
Fachhochschule

Fading Illustration with Software Defined Radio

Semester Thesis for Sara Cinzia Halter and Naoki Pross
HS 2021

1 Introduction

In real-world wireless channels a variation of signal attenuation occurs for example due to multipath propagation, shadowing effects produced by obstacles, and movement of transmitter or receiver. Channels affected by this effects are usually referred to as fading channels. To be able to make the impact on the signal visible, Software Defined Radios (SDR) come in handy, because they give a high degree of freedom regarding the signal processing part of the transmitter and receiver. This gives the possibility of interchanging parameters in the signal processing chain in software very quickly without developing new transmitter and receiver architectures.

2 Task Definition

The goal of this thesis is to develop a SDR-based demonstrator, consisting of one transmitter and one receiver, to illustrate the impact of different fading effects on the signal. To get a brief understanding of the concept of fading channels, the project should be started with a literature research followed by simulation of different scenarios, which then can be reproduced by measurements. The following tasks should be carried out:

- Evaluate a suitable development environment for the SDR (GNURadio, Matlab or others).
- Develop the signal processing chain for the SDR transmitter and receiver.
- Allow the flexibility to change parameters such as for example modulation scheme.
- Elaborate scenarios which can be demonstrated (walking, car).

3 Project Schedule & Specifications

At the beginning of the thesis, a project schedule and a specification sheet must be prepared and submitted to the supervisor within the first two weeks:

- Plan a total of 240 working hours (8 ECTS × 30 h/ECTS) per student.
- The project work should be divided appropriately within the group; the report must include a breakdown and comparison of the planned and the effectively implemented project plan.
- A typical specification sheet can be found in the Student's Guide and on the public server[1].

4 Lab Journal

Each student has to keep a handwritten and dated lab journal. The journal can be either in the form of a bound booklet or, if you have a laptop with a touchscreen, digitally but still handwritten. All activities concerning duration and results are to be recorded in it, along with the minutes of the weekly meetings. The lab journal has to be handed in at the end of the work and will be graded.

[1] Path of the public server: `\\hsr.ch\root\auw\sge\labors\Mk\pub_for_students\`

5 Thesis Report

The thesis report must contain all considerations, clarifications, calculations as well as investigations in detail (in text and figures). Beyond that, the following points need to be considered:

- The text the report should not exceed 60 pages.
- The report must be written legibly and clearly structured. Guidelines on structure and other useful information can be found in the Student's Guide, which is available in printed form in the workspace folder and digitally on the student server.
- The report must include a signed non-plagiarism declaration, an example of this explanation can also be found on the public server.
- Two printed copies of the report must be submitted. A microSD card containing all data (incl. a PDF of the report, with a file size of $\leq 5\,\text{MB}$) has to be enclosed with each report.
- Writing in English is highly encouraged, and so is the use of LaTeX.

Further details will be discussed and determined during the weekly meetings. Meanwhile, the project work should be carried out as independently as possible. The criteria for evaluation and grading are provided in the Student's Guide.

6 Important Dates

Project start:	September 20, 2021
Report submission:	December 24, 2021, 12:00
Presentation:	tbd

7 Administrative Information

Adviser:	Michel Nyffenegger, michel.nyffenegger@ost.ch
Lab assistant:	Marcel Kluser, 058 257 42 73, marcel.kluser@ost.ch
Place of work:	WsComm Lab, 2.106a
Meetings:	weekly, by arrangement, in the lab or via Teams
Examiner:	Prof. Dr. Heinz Mathis, heinz.mathis@ost.ch

Fading illustration with SDR – Project Plan

Naoki Pross, Sara Cinzia Halter

Fall semester 2021

Contents

	Date	**Signature**
Halter Sara Cinzia		
Pross Naoki Sean		

1 Introduction

For the semester thesis at the Eastern Switzerland University of Applied Sciences (OST) it has been requested to create a demonstrative setup to show the fading effect, which is present in real world wireless communication systems. The device is intended to be used for pedagogical purposes such as to show the effect at the Open Days or for demonstrations during future lectures on fading channels.

2 Task Description

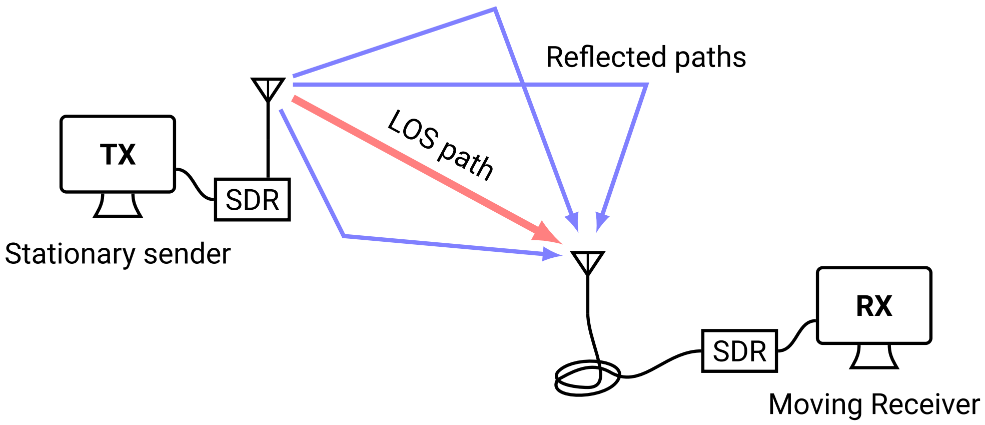


Figure 1: Sketch of the setup that will be modelled and implemented. The model will need to be adjusted depending on whether there is a line of sight (LOS) between the sender and receiver.

The scope of the project is to realize a demonstration of a fading channel using a software defined radio (SDR). Out of the many types of fading effects that exist only small scale fading effects ought to be shown in the demonstration, specifically multi path propagation fading is of interest. The project requirements that must be fulfilled are thus:

- Understand of one or more mathematical models of the fading effect.
- Evaluate a suitable development environment for the SDR.
- Develop of a signal processing chain for the SDR transmitter and receiver.
- Develop of an interface to vary the parameter of the transmission, such as the modulation scheme.
- The demonstration should work with both stationary and moving receivers.

3 Development plan

The development of the project will be carried out roughly in three phases, of which the first two will start in parallel.

1. Develop an understanding of fading and how to work with SDR devices.
2. Create a basic TX – RX line *without* a fading channel model.
3. Integrate the fading channel model into the prototype.

3.1 SDR Device

At the time of writing, we are given an *USRP B210* SDR as hardware device for the project. An evaluation based on various criteria, including software compatibility and performance, will be made to decide which combination of SDR devices and software toolboxes will used.

3.2 Prototype

The first concrete step will be the realization of a prototype both simulated and with the hardware. A TX – RX line with an ideal channel[1] will be used to test the modulation and demodulation steps of the signal processing chain. Furthermore in this phase a software interface to select the modulation scheme and to configure the modulation parameters will be developed.

3.3 Fading channels

Once the prototype works we will integrate one or more fading channel models, both in software (simulation) and in hardware. The latter will be done by taking out the devices around the campus. Some effects of the channel will require the receiver to move, walking will be used to demonstrate those a scenarios. If there will be some time left, we may also use a car to demonstrate some other effects that are not observable at a walking speed.

[1] In hardware that means a short coaxial cable between the two SDR devices.

4 Milestones

Table 1: Milestones of the project

Name	Due date	Description
Project plan	Week 40	Finalization of this document.
Working SDR TX – RX	Week 44	Completion of an RX – TX line on SDR with variable parameters for configuration.
Working fading TX – RX	Week 49	Both the simulated and the physical transmission lines work and it is possible to observe the consequences of fading.
Abstract	17 Dec. 2021	The abstract is handed in.
Documentation	Week 50	The documentation is complete both on the theory and practical sides.
Presentation	23 Dec. 2021	Presentation of the project on Campus.
Submission	24 Dec. 2021	—

Figure 2: Project schedule planned (Gantt diagram)

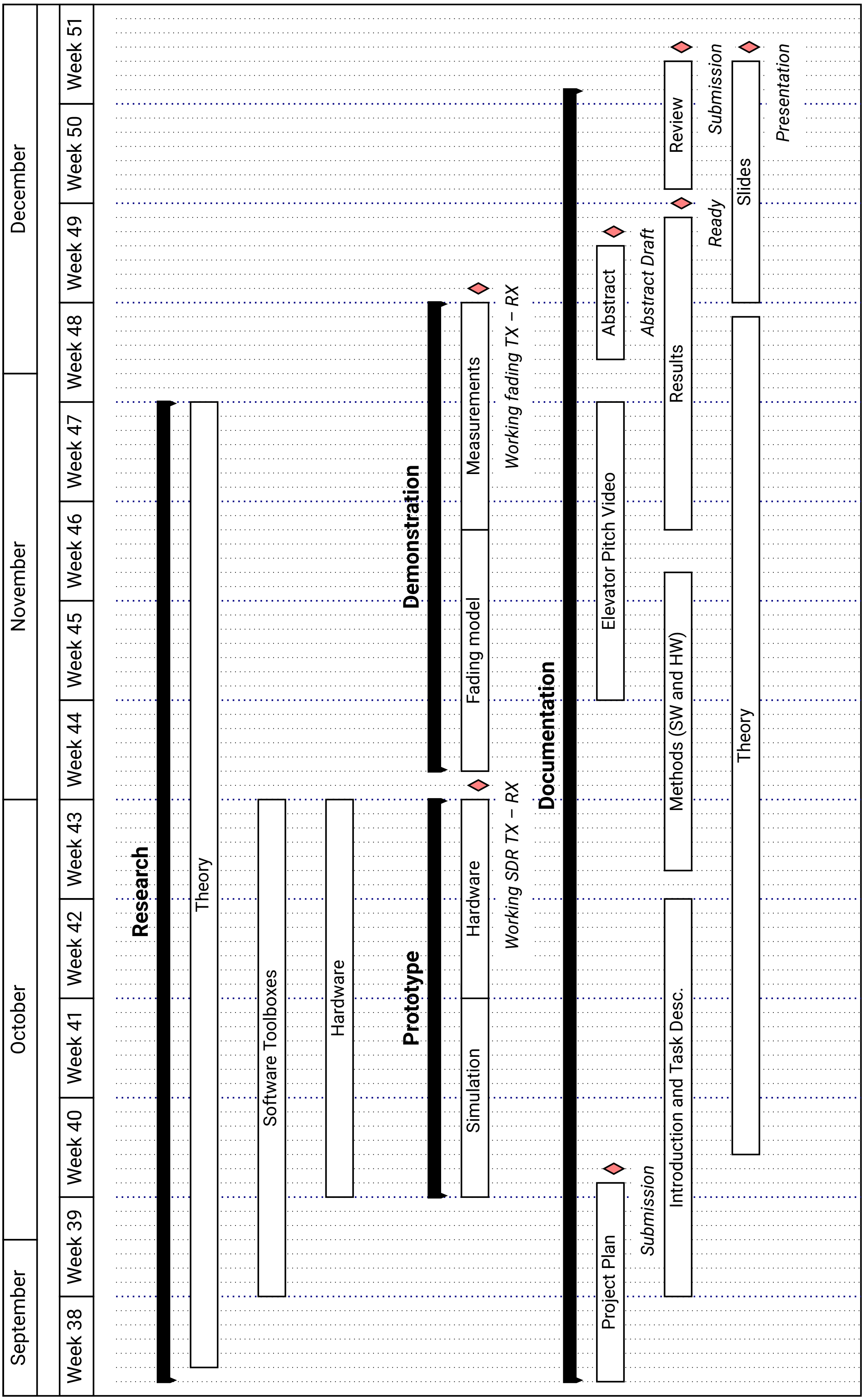
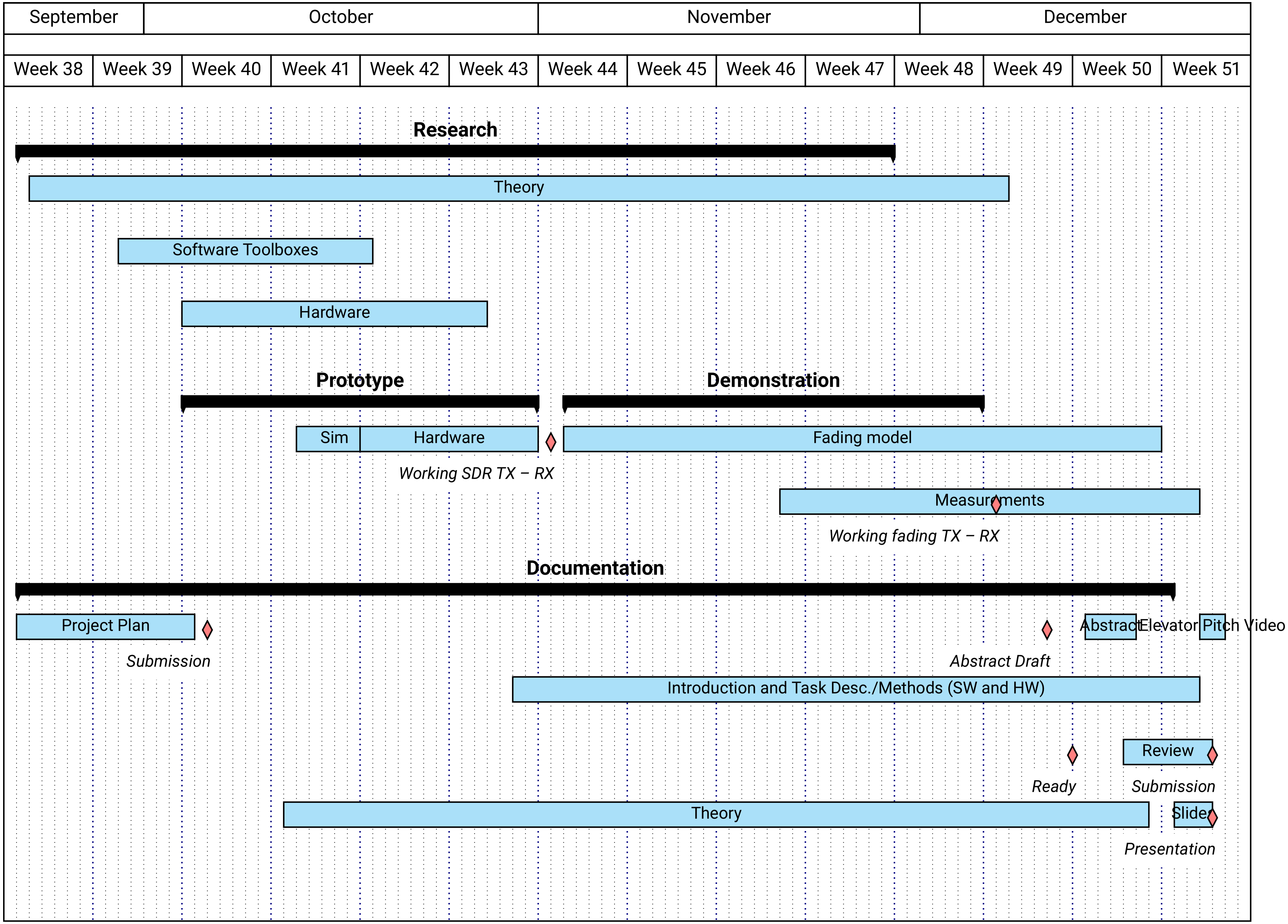


Figure 3: Project schedule effective (Gantt diagram)


September
October
November
December
Week 38
Week 39
Week 40
Week 41
Week 42
Week 43
Week 44
Week 45
Week 46
Week 47
Week 48
Week 49
Week 50
Week 51
Research
Theory
Software Toolboxes
Hardware
Prototype
Demonstration
Sim
Hardware
Fading model
Working SDR TX – RX
Measurements
Working fading TX – RX
Documentation
Project Plan
Submission
Abstract Draft
Abstract
Elevator Pitch Video
Introduction and Task Desc./Methods (SW and HW)
Review
Ready
Submission
Theory
Slides
Presentation