Multichannel coherent receiver on the RTL-SDR

Mikko Laakso

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 15.2.2019

Supervisor

Prof. Ville Pulkki

Advisor

Prof. Risto Wichman

Aalto University
School of Electrical
Engineering

Aalto University
School of Electrical
Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Mikko Laakso

Title Multichannel coherent receiver on the RTL-SDR

Degree programme Computer, Communication and Information Sciences

Major Acoustics and Audio Technology **Code of major** ELEC3030

Supervisor Prof. Ville Pulkki

Advisor Prof. Risto Wichman

Date 15.2.2019 **Number of pages** 64+1 **Language** English

Abstract

A low cost multichannel coherent SDR receiver is studied in this thesis. The proposed design is based on the affordable RTL-SDR USB dongle. The synchronization method applied utilizes a common reference signal to coincide the sample streams from RTL-SDR receivers. The maximum number of channels successfully synchronized at 1 MHz sampling rate is 35. Furthermore, adding more channels is straightforward due to the modular structure of the system. Phase-coherent state can currently be reached up to approximately 1.5 GHz, or 85% of the tuning range of the dongle.

A prototype receiver is evaluated by experimenting with MIMO space-time coding and direction of arrival estimation. Space-time coding was performed with differential block codes of dimensions 2×2 and 3×3. The transmissions are successfully received, displaying a clear BER performance increase as the number of receive antennas is increased to a maximum of 18.

Direction of arrival estimates apply the MUSIC technique on signals from a 9-element uniform linear array. The results are promising, suggesting the receiver concept could be employed in low budget MIMO research.

Keywords MIMO, software-defined radio, RTL-SDR, antenna array

Aalto University
School of Electrical
Engineering

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi
Diplomityön tiivistelmä

Tekijä Mikko Laakso

Työn nimi Monikanavainen vaihekoherentti RTL-SDR vastaanotin

Koulutusohjelma Computer, Communication and Information Sciences

Pääaine Akustiikka ja audioteknologia **Pääaineen koodi** ELEC3030

Työn valvoja Prof. Ville Pulkki

Työn ohjaaja Prof. Risto Wichman

Päivämäärä 15.2.2019 **Sivumäärä** 64+1 **Kieli** Englanti

Tiivistelmä

Tässä diplomityössä tutkitaan matalan kustannuksen monikanavaista vaihekoherenttia ohjelmistoradiota. Ehdotettu ratkaisu perustuu edulliseen RTL-SDR USB-vastaanottimeen. Työssä käytetty metodi vastaanottimien synkronointiin pohjautuu jaettuun yhteiseen referenssisignaaliin. Synkronoitujen kanavien maksimimäärä 1 MHz näytteenottotaajuudella on 35 ja järjestelmän modulaarisen rakenteen ansiosta kanavien määrää on helppo lisätä. Vaihekoherenssi saavutetaan noin 1.5 GHz taajuudelle asti. Tämä on noin 85% RTL-SDR vastaanottimen viritystaajuuden maksimista.

Prototyyppivastaanottimen toimintaa testataan MIMO aika-tila-koodatulla tiedonsiirrolla ja signaalin suunnan estimoinnilla. Työssä käytetään differentiaalisia 2×2 ja 3×3 aika-tila-koodeja. Lähetteet vastaanotetaan onnistuneesti, bittivirheiden tason vähentyessä kun vastaanottimen kanavien määrää lisätään.

Signaalin suunnan estimointiin käytetään MUSIC metodia 9-elementtisellä tasajakoisella antenniryhmällä. Saadut tulokset ovat lupaavia, ja kehitettyä vastaanotinta voitaisiin mahdollisesti jatkossa käyttää pienen budjetin MIMO tutkimuksessa.

Avainsanat MIMO, ohjelmistoradio, RTL-SDR, antenniryhmä

Preface

This research was conducted at the Department of Signal Processing and Acoustics in Aalto University School of Electrical Engineering. I would like to thank my advisor Professor Risto Wichman for his guidance and for giving me the opportunity to work in this project. I would also like to thank my supervisor Professor Ville Pulkki.

I want to thank my parents for all their support. I am grateful to my wife for her love and her patience during my studies. I also must express my gratitude to my uncle, Kari Vehosalo, for igniting my interest in technology ever since childhood.

My son, Anton, I am delighted for the joy you bring to us - may your path be smoother.

Otaniemi, 15.2.2019

Mikko Laakso

Contents

Abstract 3

Abstract (in Finnish) 4

Preface 5

Contents 6

Symbols and abbreviations 8

1 Introduction 10

2 Theoretical background 12
2.1 Quadrature modulation . . . 13
2.1.1 Phase-shift keying . . . 14
2.1.2 Differential phase-shift keying . . . 15
2.2 Matched filter . . . 15
2.3 MIMO . . . 17
2.3.1 Channel model . . . 17
2.3.2 Channel estimation . . . 19
2.4 Space-time coding . . . 20
2.4.1 Alamouti code . . . 21
2.4.2 Differential space-time codes . . . 21
2.5 Direction of arrival estimation . . . 24

3 Research material and methods 27
3.1 RTL-SDR . . . 27
3.2 FL2K VGA-adapter . . . 29
3.3 Computational methods . . . 32
3.3.1 Cross-correlation . . . 32
3.3.2 Discrete root-raised cosine filters . . . 33
3.3.3 Symbol timing synchronization . . . 34

4 Receiver design 36
4.1 Hardware . . . 36
4.1.1 Clock generation . . . 37
4.1.2 Reference noise properties . . . 39
4.1.3 Zener diode noise generator . . . 40
4.2 Software . . . 44
4.2.1 Network protocol . . . 45
4.2.2 Fractional sample index timing . . . 47
4.2.3 Time synchronization . . . 50
4.2.4 Phase correction . . . 51

5 Results **53**
5.0.1 Software performance 53
5.0.2 Phase and timing synchronization 54
5.0.3 Direction of arrival measurements 56
5.0.4 Differential unitary space-time modulation experiments 57

6 Summary **61**

References **62**

A Circuit diagram of noise- and clock generators **65**

Symbols and abbreviations

Symbols

B	bandwidth
C	capacity
f_s	sampling rate
$\mathbf{H}$	channel matrix
$\mathbf{I}$	identity matrix
T	sampling period
T_s	symbol period
η	spectral efficiency
λ	wavelength

Operators

$\mathbf{A}^{\mathrm{T}}$	Transpose of matrix $\mathbf{A}$
$\mathbf{A}^{\dagger}$	Hermitian transpose of matrix $\mathbf{A}$
$\lvert\mathbf{A}\rvert$	Determinant of matrix $\mathbf{A}$
$\mathrm{Tr}(\mathbf{A})$	Trace of matrix $\mathbf{A}$
$\mathrm{ReTr}(\mathbf{A})$	Real part of the trace of matrix $\mathbf{A}$
$d(\mathbf{a}, \mathbf{b})$	Euclidean distance of vectors $\mathbf{a}$ and $\mathbf{b}$
$\langle \mathbf{a}, \mathbf{b} \rangle$	Inner product of vectors $\mathbf{a}$ and $\mathbf{b}$
$\mathbf{a} \odot \mathbf{b}$	Element-wise product of vectors $\mathbf{a}$ and $\mathbf{b}$
$\mathrm{E}[X]$	Expected value of variable X
$\arg\max$	Argument of the maximum
$\angle a$	Angle of complex variable a
a^*	Complex conjugate of variable a

Abbreviations

AWGN	Additive white Gaussian noise
BER	Bit error rate
CSI	Channel state information
CSIT	Channel state information at the transmitter
CSIR	Channel state information at the receiver
CRLB	Cramer-Rao lower bound
DAC	Digital-to-analog converter
DDS	Direct digital synthesis
DFT	Discrete Fourier transform
DOA	Direction of arrival
DUSTM	Differential unitary space-time modulation
DVB-T	Digital video broadcasting, terrestial
FFT	Fast Fourier transform
IF	Intermediate frequency
ISI	Intersymbol interference
LO	Local oscillator
MIMO	Multiple-input, multiple-output
MMIC	Monolithic microwave integrated circuit
MUSIC	Multiple signal classification
NCO	Numerically controlled oscillator
PLL	Phase-locked loop
PSK	Phase-shift keying
RF	Radio frequency
SDR	Software-defined radio
SNR	Signal-to-noise ratio
SISO	Single-input, single-output
TX	Transmitter
ULA	Uniform linear array
USB	Universal serial bus
VGA	Video graphics array

1 Introduction

The need for capacity in wireless communication is constantly growing, with no end in sight. On the other hand, the available radio spectrum is a scarce resource. These factors have increased interest in research and implementation of multiple-input multiple-output (MIMO) communication technology. A MIMO system employs multiple transmit and receive antennas and is able to expand the capacity of a given channel compared to a single antenna link. Current communication standards, such as 4G LTE and WiFi already adopt MIMO technology. [1]

Software-defined radio (SDR) provides a flexible platform for rapid verification of communication methods in reality. By extension, an array of phase-coherent SDR can act as a testbed for MIMO system development. Commercial SDR solutions offering MIMO capability are very expensive; the price category is often in the range of thousands of euros per added receiver channel [5]. Regularly, such systems are assembled from expandable discrete SDRs synchronized to a common external clock signal. An example of such system is the Lund University Massive MIMO testbed with 100 base station antennas, built from Ettus Research USRP SDRs [6, 5]. However, the high cost of such systems shuts out low budget researchers and enthusiastic amateurs.

Aforementioned sparks motivation to seek a cost-effective alternative. This thesis presents a design for an economical phase-coherent SDR receiver, constructed from commonly available components. The material costs of the proposed design are a fraction of the cost of a multiple-USRP setup. The obvious drawback is that the technical specifications are not quite comparable, and furthermore, the design lacks transmitter capability.

The RTL-SDR, originally developed as a USB DVB-T tuner, later found use as a software-defined radio. The RTL-SDR tuning range extends to 1.7 GHz and the device supports sampling rates up to 2.56 MHz [7]. In this work, a coherent receiver based on the inexpensive RTL-SDR is developed. Stated more generally, the objective is to synchronize a large array of standalone low-end consumer-grade sensors. A generic RTL-SDR compatible USB dongle can be purchased online for less than 10€.

The relevance of a multichannel phase-coherent receiver is by no means limited to communications research. Other possible uses are in the field of remote sensing and direction finding. One could envision the system being utilized in experiments for passive radar [8], direction of arrival (DOA) or beamforming, to name a few applications.

A considerable challenge in bringing the receivers to time-synchronized state is the asynchronous nature of USB communication: the launch of the sample data streams occurs nondeterministically. The time difference can be in the order of thousands of samples. Earlier work [8, 9] suggested using common clock and reference signals for measuring and correcting this time difference. Furthermore, phase correction coefficients for each signal channel can be computed from this common reference.

In this work, an external clock generator is designed to facilitate synchronous sampling. The clock souce is a well-known Pierce-Gate oscillator circuit [10], buffered

to multiple outputs. As for the reference signal, a wideband noise generator is constructed. The source of the wideband noise is a reverse-biased Zener diode [11, 12, 13]. The small noise voltage produced by the diode is amplified through a chain of broadband MMIC amplifiers to multiple output stages. The reference noise is distributed to the signal receivers via transmission line coupling effect, i.e. the antenna inputs are galvanically isolated from the reference generator. The design proposes a coupler module, which accommodates seven RTL-SDR dongles. That part of the design was set, as a printed circuit board designed by Peltola [9] to distribute the reference signal to seven signal receivers, was already available.

As the computational load of real-time signal processing is high in this task, the software for synchronizing the sample streams was written in C++. The software compiles as a command line utility for Linux. The phase aligned samples are published to a network socket, to which client software consuming the samples connects. To continue with the client-server paradigm, a rudimentary protocol for controlling the receiver software remotely was implemented.

The objective is met and the end result could legitimately be termed a massive coherent array on the RTL-SDR. At the time of writing, a maximum of 35 signal streams have been successfully synchronized. Further expansion of the receiver is relatively easy, but requires some assembling, therefore this task is left for future work.

The functionality of the system is empirically verified by wireless communication experiments over a $3 \times N$ MIMO channel in the frequency range of 1 GHz. The testing is conducted in a typical office environment, at quite close range, because the available transmitter power is insignificant. An inexpensive USB to VGA adapter [14] acts as a transmitter in all these experiments. The scope here is limited to differential coding schemes, which permit non-coherent detection. Differential space-time coding sacrifices performance slightly to reduce the detector complexity at the receiver. Coherent detection, which requires channel state information (CSI) [1, 2] to be estimated, was excluded from the scope.

Finally, direction of arrival (DOA) is tested with the popular Multiple Signal Classification (MUSIC) [15] algorithm. The algorithm is based on the spectral decomposition of the signal covariance matrix and yields a more accurate estimate than DFT based methods. This requires a phase-coherent receiver while the differential space-time coding experiment does not.

The structure of this thesis is as follows. The first chapter outlines the relevant digital communication theory, starting from a single antenna link. In the second chapter, functionality of the RTL-SDR and the FL2K VGA adapter are examined, along with some of the relevant signal processing methods used. Next chapter deals with the design of the hardware and software needed to synchronize the RTL-SDR dongles. The results of the experiments are shown in Chapter 5. Finally, a short summary is given last in Chapter 6.

2 Theoretical background

This chapter outlines the theory of wireless transmission with digital modulations. Before considering multiple antenna communication, it makes sense to examine the regular single antenna link, also known as a single input single output (SISO) system. Here, only linear modulation is considered. Nonlinear modulation methods such as frequency shift keying are not covered.

MIMO systems employ arrays of antennas to increase the channel capacity and by consequence, spectral efficiency. In a conventional single-input single-output system, the Shannon limit dictates the maximum achievable capacity in bits per second as a function of bandwidth and SNR

$$C = B \log_2 \left(1 + \frac{P}{\sigma^2}\right) \tag{1}$$

where B is the bandwidth in Hz and P the received signal power. Equation (1) sets the upper bound for bit rate, at which information can be transmitted such that the probability of error is arbitrarily small [2, p. 7]. A MIMO system can overcome this limit, without increasing the power or bandwidth, by employing spatial multiplexing [18, p. 19]. In Section 2.3 it is shown how diversity techniques can increase the reliability of the link, without increasing bandwidth.

The performance of a communication system is in most cases modeled assuming an additive white Gaussian noise (AWGN) environment [16, pp. 202-208]. An AWGN environment implies that the noise process is zero mean and has equal power across all frequencies. In other words, noise spectral density N_0 is constant. In a sampled system, this relates to the noise variance σ^2 by

$$\sigma^2 = \frac{N_0}{2T} \tag{2}$$

where T is the sample time [16, pp.202-208]. It is useful to relate the sample SNR to energy per symbol to noise power spectral density E_S/N_0. Almost invariably, there are several samples per symbol $T_s > T$. Relating the symbol energy to noise energy in the bandwidth, this quantity is given by

$$\frac{E_S}{N_0} = \frac{P_s T_s}{N_0 B_n T} \tag{3}$$

where P_s is the signal power, T_s is the symbol time and B_n the noise bandwidth. For complex samples, the noise bandwidth is equal to the sampling rate, $B_n = f_s$. The commonly used measure of signal quality E_b/N_0, energy per bit to noise spectral density, is obtained from (3) by simply dividing with the number of bits per symbol. [17, pp. 5-6]

A MIMO link still consists of multiple passband channels, which will need to be translated to and from baseband symbols. Therefore, quadrature modulation and pulse shaping are introduced.

2.1 Quadrature modulation

Frequency translation between baseband and passband is achieved by modulating the baseband signal with a carrier. Multiplying an information bearing signal $s(t)$ and a carrier signal at angular frequency ω_c shifts the frequency content of $s(t)$ by ω_c. If the Fourier transform of the signal is $S(\omega) = \mathcal{F}\{s(t)\}$, then the Fourier transform pairs

$$\begin{aligned} s(t)e^{j\omega_c t} &\longleftrightarrow S(\omega - \omega_c) \\ s(t)\cos(\omega_c t) &\longleftrightarrow \frac{1}{2}[S(\omega - \omega_c) + S(\omega + \omega_c)] \end{aligned} \tag{4}$$

show the relation for complex $e^{j\omega_c t}$ and real sinusoidal carriers. In this context, the multiplication operation is also referred to as mixing. In the case of a real carrier signal, the multiplication produces two half-amplitude images of the spectrum, known as sidebands in RF terms. The negative frequency lower sideband is reflected to $|\omega - \omega_c|$ in the positive side of the spectrum [16, p. 704]. The full complex product is difficult to produce in continuous time systems, furthermore, it would require two independent channels to transmit the complex result. In local discrete time processing, full complex modulation does not pose a problem [16, p. 707].

M-ary linear digital modulation can be viewed as a projection from M symbol waveforms to signal space. The transmitted information is mapped to symbols, where each symbol represents a $\log_2(M)$-bit sequence [16, p. 214]. In quadrature modulation, the signal basis functions are two sinusoids with a phase difference of $\pi/2$. These are referred to as in-phase and quadrature components. In continuous time, the passband signal of a quadrature modulator is written in terms of cosine and sine carriers as

$$s(t) = x_I(t)\cos(\omega_c t) - x_Q(t)\sin(\omega_c t) = \mathrm{Re}\{x(t)e^{j\omega_c t}\} \tag{5}$$

where ω_c is the carrier frequency. $x_I(t)$ and $x_Q(t)$ are the baseband pulse waveforms and for the rightmost expression, their sum $x(t) = x_I(t) + jx_Q(t)$. The basis is orthogonal, since $\langle cos(\cdot), sin(\cdot)\rangle = 0$ for any integer multiple of periods. [17, pp. 27-30]

The quadrature demodulation from passband to baseband is performed in the receiver by multiplying with two sinusoids 90° out of phase, as a complex exponential $e^{j\omega_c t}$. Expanded from exponential to Cartesian form with the help of trigonometric identities it reads

$$\begin{aligned} y(t) = s(t)e^{-j\omega_c t} &= \frac{1}{2}\left(x_I(t) + jx_Q(t)\right) \\ &+ \frac{1}{2}\left[(x_I(t) - jx_Q(t))\cos(2\omega_c t) - (x_Q(t) + jx_I(t))\sin(2\omega_c t)\right] \end{aligned} \tag{6}$$

The double frequency terms on the second line of (6) are removed by filtering in real-world applications. The signal that the modulation (5) produces is real, but using the

complex representation to create the signal simplifies analysis and implementations. Although it can arguably be viewed as a mathematical construct. If either of the inputs to quadrature modulation are zero, $x_Q(t) = 0$ or $x_I(t) = 0$, (5) reduces to ordinary amplitude modulation. Likewise, the demodulation (6) can be used in the detection of traditional amplitude modulation, or any analog modulation for that matter. Still, since only the real part of the complex modulation is retained in (5), the spectrum is two-sided around ω_c [16, p. 707].

The advantage that the quadrature demodulation has is the ability to discern negative and positive frequencies. In discrete time system sampled at rate f_s, this implies the observable bandwidth extends from $-f_s/2$ to $f_s/2$.

2.1.1 Phase-shift keying

Phase-shift keying is a modulation technique where the information is encoded into variation of carrier phase. The amplitude of the carrier does not vary, thus the baseband pulse waveforms are all of equal magnitude. The set of baseband pulse waveforms that comprises a modulation scheme are often visualized with the help of a two-dimensional constellation diagram. The constellation diagrams of unrotated binary phase-shift keying, quadrature-phase shift keying and 8-psk are depicted in Figure 1. BPSK is a special case with a real constellation, since $x_Q = 0$ and as such it would not require a quadrature carriers.

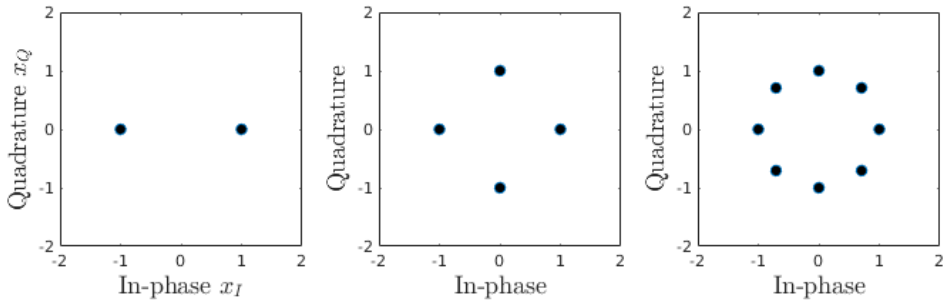

Figure 1: Constellations for BPSK (left), QPSK (middle) and 8-PSK (right).

If the pulse shape of $x(t)$ is rectangular, the resulting signal from (5) would change phase abruptly, yielding unwanted high-frequency content. Therefore, matched filter pulse shaping, which limits the support of $x(t)$ between $[-T_s/2, T_s/2]$ is employed in both continuous and discrete implementations to decrease the bandwidth.

Consider a discrete time receiver operating with a local oscillator ω_c, with an unknown phase offset θ w.r.t the received signal. Further, assuming ideal symbol timing estimate, such that the signal is sampled at exactly the symbol rate of the transmitter. For analysis, the received baseband waveform filtered with pulse shaping filter h_{mf} can be written

$$y(nT) = |h_{11}|e^{j\theta} \sum_k a_k h_{mf}(nT - kT_s)e^{-j\omega_c} + w(nT) \tag{7}$$

where $h_{11} = |h_{11}|e^{j\theta}$ is a complex fading coefficient and $w(nT)$ an identically distributed complex Gaussian random variable describing the noise in the system. To be able to reliably detect the transmitted symbols a_k, the phase offset must be estimated at the receiver [16, pp. 360-362].

2.1.2 Differential phase-shift keying

Differential phase-shift keying (DBPSK) resolves the phase ambiguity by encoding and decoding the current symbol in relation to the previous symbol. Now considering baseband symbols, the encoding and decoding is written

$$\begin{aligned} a'_k &= a'_{k-1}a_k \\ y'_k &= y^*_{k-1}y_k \end{aligned} \tag{8}$$

where a_k is a symbol drawn from an M-ary constellation, a'_k is the differentially coded symbol and y_k the kth received symbol before differential decoding [16, pp. 398-399]. The initial transmitted symbol a_0 could be any of the symbols from the constellation, but it is usually set to the first symbol. Similarly, at the decoder y_0 an is arbitrary symbol from the respective constellation.

Following is an example for M-ary DPSK, e.g. mapping information bits to symbols from Fig. 1. Now if the receiver encodes two $\log_2(M)$ codes z, $0 <= z <= M-1$, at the detector we have

$$\left(e^{j\theta}e^{\frac{j2\pi z_1}{M}}\right)^* \left(e^{j\theta}e^{\frac{j2\pi z_1}{M}}\right)\left(e^{j\theta}e^{\frac{j2\pi z_2}{M}}\right) = e^{\frac{j2\pi z_2}{M}} \tag{9}$$

The phase offset, assumed to remain constant for the duration of the two symbols, vanishes. The cost is that differential modulations generally require 3dB more SNR compared to nondifferential modulation to achieve the same performance.

2.2 Matched filter

Matched filters are used in communications to maximize the signal-to-noise ratio at the sampling instant, in the presence of additive noise. The filtering is split between the transmitter and the receiver, each applying complementary filters. The filter also serves to limit the bandwidth of the signal and to reduce intersymbol interference. Considering the bandwidth in baseband signaling, given that there was infinite bandwidth available, a signal could transition instantly between symbols.

If the pulse shape function has finite support in time domain, it will have infinite support in frequency domain. Conversely, the frequency domain finite support transforms to infinite support in time domain. By intersymbol interference (ISI) we refer to the situation where energy from other symbols in the sequence disrupts the detection of the current symbol. Thus, the time domain support of the pulse shape function should be short. Either that, or the pulse shape should be zero valued at integer multiples of symbol time T_s. The Nyquist ISI criterion describes a channel is not affected by ISI. [16, pp. 686-686]

When the pulse is filtered with the matched filter h_a in the transmitter, and again with the complementary matched filter h_b in the receiver, the overall shape of the combined filter is the convolution [16, pp. 686-686]

$$h_{mf}(\tau) = \int_{-\infty}^{\infty} h_a(t)h_b(t-\tau)dt \tag{10}$$

The matched filter output at the receiver for the k:th symbol is

$$y(kT_s) = \sum_i a_i h_{mf}((k-i)T_s) = a_k + \sum_{i \neq k} a_i h_{mf}((k-i)T_s) \tag{11}$$

where the desired symbol is a_k and the sum term represents the contribution of other symbols in the transmitted sequence.

The matched filter in the receiver is the time reversed and conjugated version of the transmitter pulse shape, $h_b(t) = h_a(-t)^*$. In the case that the shape is a real-valued even function, $h_b(t) = h_a(t)$. The Nyquist ISI criterion for no ISI is defined by

$$h_{mf}(kT_s) = \begin{cases} 1 & k = 0 \\ 0 & k \neq 0 \end{cases} \tag{12}$$

The well-known raised cosine pulse shape is perhaps the most used pulse shape fulfilling the criterion. In frequency domain, the shape of the response is defined by

$$|H_{rc}(f)| = \begin{cases} T_s & 0 \leq |f| \leq \frac{1-\beta}{2T_s} \\ \frac{T_s}{2}\left[1 + \cos\left(\frac{\pi|f|T_s}{\beta} - \frac{\pi(1-\beta)}{2\beta}\right)\right] & \frac{1-\beta}{2T_s} \leq |f| \leq \frac{1+\beta}{2T_s} \\ 0 & |f| > \frac{1+\beta}{2T_s} \end{cases} \tag{13}$$

where $0 \leq \beta \leq 1$ is the roll-off factor. The roll-off factor determines the excess bandwidth of the filter. When $\beta = 0$, response (14) approaches the ideal rectangular brick-wall filter response. In time-domain, the impulse response reduces to the *sinc*() function, which has infinite support. When $\beta = 1$, the frequency response shape is the true raised cosine. The time domain response is shorter in this case.

The continuous time response of the raised cosine (13) is

$$h_{rc}(t) = \frac{\sin\left(\frac{\pi t}{T_s}\right)}{\frac{\pi t}{T_s}} \cdot \frac{\cos\left(\frac{\pi\beta t}{T_s}\right)}{1 - \left(\frac{\pi\alpha t}{T_s}\right)^2} \tag{14}$$

To split the filtering between the transmitter and receiver, an inverse Fourier transform of $\sqrt{H_{rc}}$ yields the root raised cosine. The continuous time response of the root raised cosine is

$$h_{rrc}(t) = \frac{1}{\sqrt{T_s}} \frac{\sin\left(\pi(1-\beta)\frac{t}{T_s}\right) + \frac{4\beta t}{T_s}\cos\left(\pi(1+\beta)\frac{t}{T_s}\right)}{\frac{\pi t}{T_s}\left(1 - \left(\frac{4\beta t}{T_s}\right)^2\right)} \tag{15}$$

In (15), the denominator is zero at $t = 0$ and $t = \pm T_s/(4\beta)$. At these points, the response must be evaluated by taking the limits $\lim_{t\to 0} h_{rrc}(t)$ and $\lim_{t\to T_s/(4\beta)} h_{rrc}(t)$. Note that $h_{rrc}(t)$ is not zero at integer multiples of T_s as required in (12), but the convolution, raised cosine $h_{rc} = h_{rrc}(t) * h_{rrc}(t)$ is.

2.3 MIMO

In wireless communication, the term fading refers to attenuation of a signal due to obstacles, weather or destructive interference arising from multi-path propagation. For a wideband channel, the fading can also be frequency dependent. In wireless communication terminology, this is called frequency-selective fading. This work considers only narrowband channels, where a single link channel experiences similar fading over its whole bandwidth. This is referred to as flat fading. [2, pp. 54-55]

Naturally, the fading process may not remain constant over time. Quite rarely are the signal propagation conditions so lenient in real-life scenarios. Three distinct cases are categorized by the rate of change: fast fading, block fading and quasi-static fading. In fast fading, the channel is considered constant over one symbol period. Block fading means that the channel remains fixed for the time frame of multiple symbols. Quasi-static fading implies that the channel remains constant for the duration of the whole transmission. The period which the channel remains constant is also known as the coherence time. [2, pp. 54-55]

In a typical wireless communication scenario where direct line-of-sight between transmit and receive antennas does not exist, the channel matrix entries are considered to be Rayleigh distributed. When a line-of-sight exists, the entries are best described by a Rician distribution. [18, pp. 58-59]

Diversity techniques in communications means transmitting multiple copies of the same information, replicated in temporal, frequency or spatial domains. This is done to combat the effects of channel fading, with the assumption that the replicas experience different fading. The redundancy helps to increase the reliability of the link in difficult fading conditions. [2, pp. 54-55]

Time diversity methods spread delayed replicas of the information in the signal. The period at which these replicas are inserted should be greater than the channel coherence time. Space diversity, also referred to as antenna diversity, replicates the information across multiple uncorrelated spatial propagation paths. Spatial diversity has the desirable property that it does not decrease the bandwidth efficiency, unlike the other two diversity methods. [2, pp. 54-55]

Space diversity is further categorized into receive and transmit diversity. Receive diversity refers to the increased total received signal power due to combining at the receiver. Closed-loop transmit diversity, while enabling significant capacity boost, requires channel state information at the transmitter (CSIT), which implies a feedback channel for relaying CSIT back to the transmitter [2, pp. 56-57]. Open-loop space-time coding, discussed in Section 2.4, overcomes this limitation by specific code design.

2.3.1 Channel model

For analysis purposes, signal propagation is commonly modeled with a flat fading channel model, which means that the model is memoryless, quantified entirely by complex gain coefficients. In the linear model, the $n_R \times n_T$ channel matrix $\mathbf{H}$ entries describe the propagation between individual n_T transmit and n_R receive antennas, as shown in Figure 2.

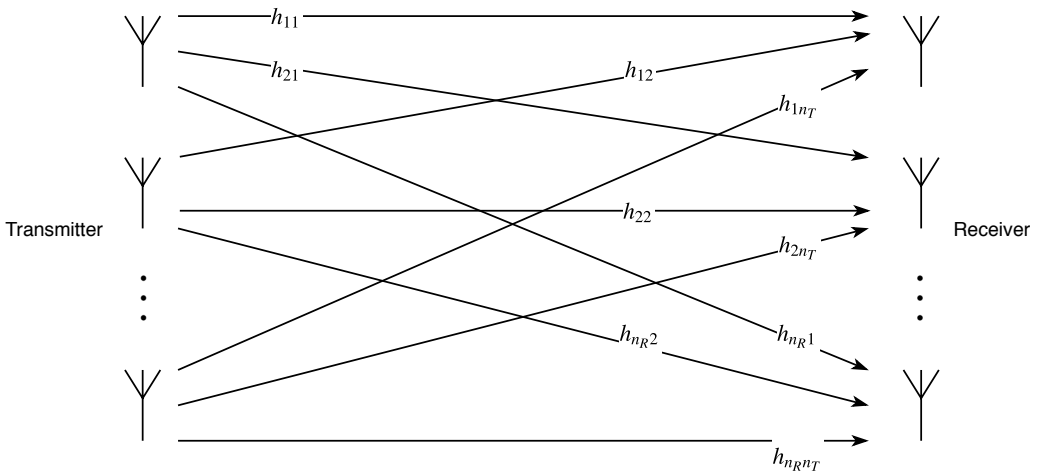

Figure 2: MIMO channel model.

Let $\mathbf{x}$ be the $n_T \times 1$ vector of transmitted signal. Then the received signal vector $\mathbf{y}$ becomes

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{w} \tag{16}$$

where $\mathbf{w}$, $n_R \times 1$, is a vector of white Gaussian noise. The noise is assumed to be independent and identically distributed, with equal variance σ^2 in the complex and imaginary parts. [2, p. 3]

The capacity of a MIMO channel can exceed the limit set by (1). If the channel $\mathbf{H}$ is known at the receiver, the capacity of the channel is ideally

$$C = B \log_2 \left| \mathbf{I}_{nR} + \frac{P}{\sigma^2 n_T} \mathbf{H}\mathbf{R}_{\mathbf{xx}}\mathbf{H}^\dagger \right| \tag{17}$$

where $|\cdot|$ denotes the determinant and the expected signal covariance is $\mathbf{R}_{xx} = \mathrm{E}[\mathbf{x}\mathbf{x}^\dagger]$. Thus, the capacity depends on the current realization of the channel matrix. For random channels, (17) should be evaluated against the expectation of $\mathbf{H}$ to yield the ergodic capacity. The ergodic capacity is visualized for few configurations with equal number of transmit and receive antennas in Figure 3.

Knowledge of the channel at the transmitter makes it possible to allocate TX power of the transmitted signals in such a way that the capacity increases. An optimal algorithm for power allocation is the "water-filling principle", where channels in good condition are assigned more TX power [18, pp. 24-25]. If CSIT is not available, all antennas at transmitter are given equal power and $\mathbf{R}_{\mathbf{xx}} = \mathbf{I}_{n_T}$, meaning $\mathbf{x}$ is statistically independent [18, p.23].

In a realistic situation, the received signals are often correlated, which reduces the link performance. However, at a maximum correlation of 0.5, the impairment to diversity gain or spatial multiplexing advantage is still negligible. The fading correlation is dependent on the antenna spacing in the receiver array, the angle of arrival and beamwidth of the incoming signal. Signals arriving from the broadside of

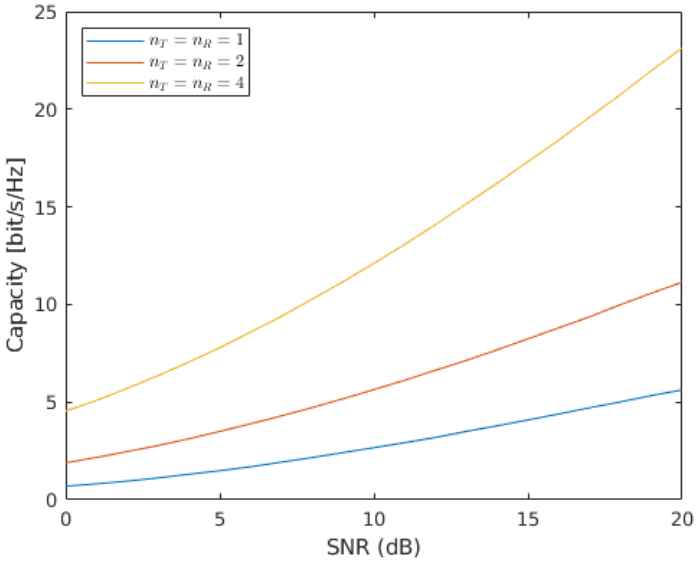

Figure 3: Ergodic capacity for MIMO systems with 1, 2 and 4 TX and RX antennas.

the array require smaller antenna spacing to achieve low correlation. The minimum distance required between the antennas to establish independent propagation paths ranges from a few wavelengths λ to the order of 10λ. [19]

2.3.2 Channel estimation

Knowledge of the channel is a prerequisite for coherent detection at the receiver. Estimating the channel matrix is made possible by transmitting known symbols along with the information symbols. The downside is, that such structures necessarily complicate the receiver structure and decrease the net capacity of the channel. The fading conditions determine how often a training symbol, or a sequence of such symbols, needs to be transmitted. The more training symbols there are, the more accurate the CSI estimate. On the other hand, these symbols can be seen as dead weight, since they do not carry useful information. [18, pp. 92-93]

When the channel experiences slow fading, it might be sufficient to prepend a frame of information symbols with a known preamble. The receiver then compares the received symbol to the original and determines CSI. The information in the rest of the frame is detected using the same channel coefficients, meaning the channel state must remain constant for the duration of the frame, consisting of the preamble and the payload. [18, pp. 92-93]

In fast fading, the channel conditions change too rapidly for the above preamble approach. Pilot symbols need to be inserted in-between the information symbols in each transmission frame. Multi-carrier modulations, such as OFDM, allow the pilots to be transmitted simultaneously on separate subcarriers [2, pp. 217-219].

The least squares estimate for the channel matrix is as follows. Assuming the training symbol, identified and correctly sampled is received in $\mathbf{Y}_k$, and $\mathbf{P}_k$ contains

the original training symbol affected by channel $\mathbf{H}_k$. The error matrix $\mathbf{E}_k^2$ is

$$\mathbf{E}_k^2 = (\mathbf{Y}_k - \mathbf{P}_k\mathbf{H}_k)^\dagger(\mathbf{Y}_k - \mathbf{P}_k\mathbf{H}_k) \tag{18}$$

The ordinary least squares solution to minimizing (18) is found with

$$\hat{\mathbf{H}}_k = (\mathbf{P}_k^\dagger\mathbf{P}_k)^{-1}\mathbf{P}_k^\dagger\mathbf{Y}_k \tag{19}$$

If the training sequence is a unitary matrix, $(\mathbf{P}_k^\dagger\mathbf{P}_k)^{-1} = \mathbf{I}$, by inserting (16), (19) simplifies to

$$\hat{\mathbf{H}}_k = \mathbf{P}_k^\dagger\mathbf{Y}_k = \mathbf{H}_k + \mathbf{W}_k \tag{20}$$

where $\mathbf{W}_k = \mathbf{P}_k^{-1}\mathbf{W}$ is an AWGN noise matrix. The least squares method is a straightforward technique to obtain $\hat{\mathbf{H}}$, but it is less robust in low SNR conditions. Minimum Mean Squares Estimation would yield more accurate results in noisy channels at the cost of increased complexity [18, pp. 93-95]. The MMSE estimate requires knowledge of the noise variance. It is given by

$$\hat{\mathbf{H}} = \mathbf{P}^\dagger(\mathbf{P}\mathbf{P}^\dagger + \sigma^2\mathbf{I}_{n_R})^{-1}\mathbf{Y} \tag{21}$$

In addition to the variance, it requires a matrix inverse of $(\mathbf{P}\mathbf{P}^\dagger + \sigma^2\mathbf{I}_{n_R})$, in $\mathbb{C}^{n_R \times n_R}$ to be computed, if implemented directly with (21).

2.4 Space-time coding

To approach the potential capacity of a MIMO channel, a practical method to exploit the diversity gains is needed. Space-time coding refers to techniques which take advantage of both space- and temporal diversity. There exists a multitude of such coding techniques, but the sections below focus only on space-time block codes (STBC), which encode the information symbols into a $n_T \times p$ matrix. The duration of one block spans p discrete transmission periods, or symbol periods. The rate of a space-time block code is defined

$$R = \frac{k}{p} \tag{22}$$

where k is the number of symbols encoded in each block. The spectral efficiency of such code is

$$\eta = RM \tag{23}$$

where again M is the size of the constellation set. For a given STBC to achieve full diversity of n_T, the rate of the code must be $R \leq 1$. Rates that fall below one expand the bandwidth by $1/R$. Consequently, the codes for which $R = 1$ are considered attractive. [2, p. 99]

2.4.1 Alamouti code

The block code introduced by Alamouti in 1998 [3] was the first to achieve full transmit diversity. The proposed scheme is an orthogonal block code for two transmit antennas, with block duration of two transmit periods. The information to be transmitted during each block is first mapped to symbols s_1 and s_2 drawn from an M-ary constellation, e.g. QPSK. Represented as a matrix

$$\mathbf{S} = \begin{bmatrix} s_1 & -s_2^* \\ s_2 & s_1^* \end{bmatrix} \tag{24}$$

where the columns of the matrix signify the transmission period and the rows index the antennas. That is, during the first interval, antenna one transmits s_1 and antenna two s_2. The signals transmitted are orthogonal, since the inner product of the rows of $\mathbf{S}$ is zero, $\langle \mathbf{s}_{1,*}, \mathbf{s}_{2,*} \rangle = s_1 s_2^* - s_2^* s_1 = 0$. [18, pp. 76-78]

As such, the Alamouti code requires coherent detection, meaning CSI needs to be estimated at the receiver. For two transmit antennas and one receive antenna, the channel $\mathbf{h}$ and the received signal $\mathbf{y}$ are 1×2 vectors. A maximum likelihood detector minimizes

$$\left|\mathbf{y} - \mathbf{h}\hat{\mathbf{S}}\right|^2 = |y_1 - h_1\hat{s}_1 - h_2\hat{s}_2|^2 + |y_2 + h_1\hat{s}_2^* - h_2\hat{s}_1^*|^2 \tag{25}$$

where $\hat{s}_1$ and $\hat{s}_2$ are checked against every possible value from the transmitted constellation. This can also be written as

$$(\hat{s_1}, \hat{s_2}) = \underset{(\hat{s_1},\hat{s_2})\in C}{\arg\min} = (|h_1|^2 + |h_2|^2 - 1)(|\hat{s_1}|^2 + |\hat{s_2}|^2) + d^2(\tilde{s_1}, \hat{s_1}) + d^2(\tilde{s_2}, \hat{s_2}) \tag{26}$$

where $\tilde{s_1} = h_1^* y_1 + h_2 y_2^*$ and $\tilde{s_2} = h_2^* y_1 - h_1 y_2^*$ and C is the constellation set [3, 2]. The decision rule can be separated into independent detectors and for PSK signals, it can be further simplified to

$$\begin{aligned} (\hat{s_1}) &= \underset{(\hat{s_1})\in C}{\arg\min} = d^2(\tilde{s_1}, \hat{s_1}) \\ (\hat{s_2}) &= \underset{(\hat{s_2})\in C}{\arg\min} = d^2(\tilde{s_2}, \hat{s_2}) \end{aligned} \tag{27}$$

The Alamouti code was later proved to be the only space-time block code that reaches full rate of 1, including full diversity of 2. The method was later extended to cases $n_R > 1$ by Alamouti [2, p.103].

2.4.2 Differential space-time codes

In SISO communications, differential phase shift keying (DPSK) resolves the phase ambiguity at the receiver by coding the information to the change of carrier phase. The detection depends on two symbols, current and the last received. It has the advantage of enabling non-coherent detection, removing the need for CSI. Noncoherent detection generally performs approximately 3 dB worse compared to coherent detection. The same holds true for differential space-time codes on multi-antenna links [4].

In 2001, Hughes [4] proposed a differential unitary group code for two transmit antennas spanning two transmission periods. This method may also be called Differential unitary space-time modulation (DUSTM). The proposed scheme can be thought of as an extension of QPSK for $n_T = 2$. All the signals in it belong to the constellation set $\mathcal{A} = \{1, -1, j, -j\}$. The code group is

$$\mathcal{G} = \left\{ \pm \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \pm \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \pm \begin{bmatrix} j & 0 \\ 0 & -j \end{bmatrix}, \pm \begin{bmatrix} 0 & j \\ j & 0 \end{bmatrix} \right\} \tag{28}$$

Since the cardinality of the code group $|\mathcal{G}| = 8$, each matrix can encode a 3-bit code word $z_t \in \{0, 1, ..., |\mathcal{G}|\}$. The rate of the code, using (22) is $\log_2 |\mathcal{G}|/p$=1.5. The group consists of unitary matrices, thus for each code k in $\mathcal{G}$, $\mathbf{G}_k^\dagger \mathbf{G}_k = \mathbf{I}_2$. The transmission is encoded differentially with the previous and current transmission matrix $\mathbf{G}_{z_t}$ [4, 2]

$$\mathbf{X}_t = \mathbf{X}_{t-1} \mathbf{G}_{z_t} \tag{29}$$

Transmission begins with the initial matrix $\mathbf{X}_0 = \mathbf{D}$,

$$\mathbf{D} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \tag{30}$$

for which $\mathbf{D}^\dagger \mathbf{D} = 2\mathbf{I}_2$. The result of the differential coding preservers the constellation $\mathbf{D}\mathcal{G} \in \mathcal{A}^{2\times 2}$. In other words, the constellation does not expand [4]. The maximum likelihood detector for the code is given by

$$\begin{aligned} \hat{\mathbf{G}} &= \arg\max_{G \in \mathbf{G}} \mathrm{ReTr}(\mathbf{Y}_{t-1} \mathbf{G}_z \mathbf{Y}_t^\dagger) \\ &\quad \arg\max_{G \in \mathbf{G}} \mathrm{ReTr}(\mathbf{G}_z \mathbf{Y}_t^\dagger \mathbf{Y}_{t-1}) \end{aligned} \tag{31}$$

The term $\mathbf{Y}_{t-1} = \mathbf{H}\mathbf{X}_{t-1} + \mathbf{W}_{t-1}$ in (31) can be seen as the channel estimate for a correlation receiver [4].

The pairwise error probability for this code is bounded by

$$\Pr\{\hat{\mathbf{G}}_k \to \hat{\mathbf{G}}_n\} \leq \frac{1}{\left| \mathbf{I} + \frac{\rho_t^2 \tilde{n}}{8(1+\rho_t \tilde{n})} (\mathbf{G}_k - \mathbf{G}_n)(\mathbf{G}_k - \mathbf{G}_n)^\dagger \right|^{n_R}} \tag{32}$$

where $\rho_t = \rho/n_T$ is the SNR per receive antenna and $\tilde{n} = 2n_T$ [4]. However, deriving (32) is out of the scope of this work. Figure (4) depicts the frame error rate for varying number of receive antennas. It is the upper bound for the probability that the detector (31) chooses an incorrect code, assuming antennas are perfectly uncorrelated.

Next, we consider extending the previous 2×2 code to three transmit antennas, since the transmitter used in the experiments is capable of transmitting a maximum of three simultaneous signals. The following discusses a unitary group code of 3×3 matrices.

In a group in $\mathbb{C}^{N\times N}$, there can be N^2 trace-wise orthogonal matrices [20]. Trace-wise orthogonality is defined w.r.t trace inner-product, such that $\mathrm{Tr}(\mathbf{G}_k^\dagger \mathbf{G}_n) = 0$

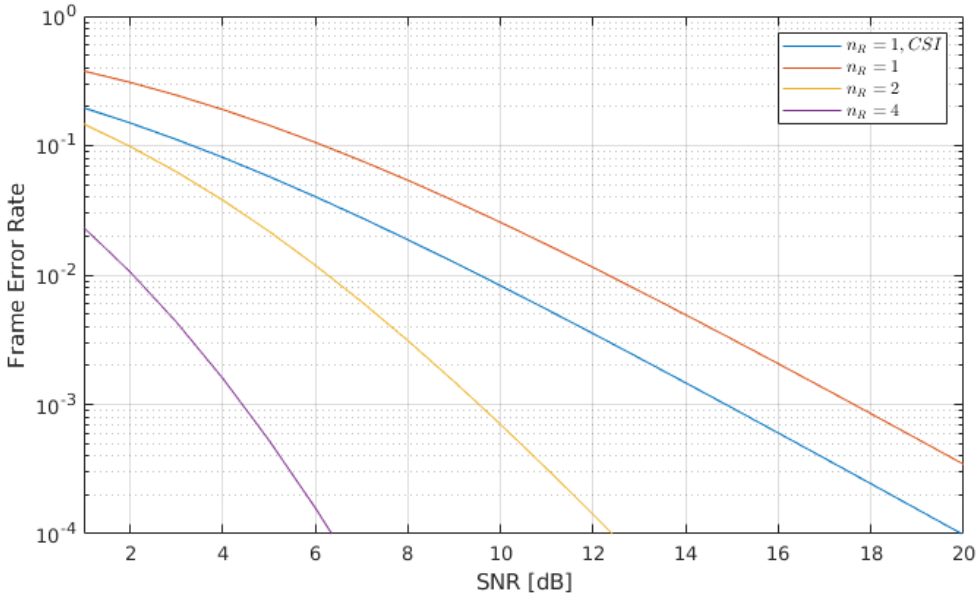

Figure 4: Pairwise error probability with 1, 2 and 4 receiver antennas. For comparison, $n_R = 1$ plotted with perfect CSI.

when $n \neq k$. That is, only the dot products of the corresponding column vectors of $\mathbf{G}_k$ and $\mathbf{G}_n$ are required to be zero.

Akin to the 2×2 code above, the cardinality of the code group can be doubled by extending it with the negated basis group. Considering the detector, this will cause the trace of the mirrored code to be equal in magnitude, but with a negative sign. Because the modulation is differential, this choice does not result in ambiguity in the absence of CSI. The codes in the group are still uniquely identifiable.

Proper Hadamard matrices are by definition square, of order $2n$, with entries restricted to $\{-1, 1\}$. If complex entries are allowed, Hadamard-like matrices of odd orders exist. For $n = 3$, such matrix is

$$\mathbb{H}_3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & \sigma^2 & \sigma \\ 1 & \sigma & \sigma^2 \end{bmatrix} \ , \ \sigma = e^{2\pi j/3} \tag{33}$$

Notice that the above is the 3-point DFT matrix [21]. Various unitary bases have been developed in the theoretical physics community. Schwinger basis yields a n^2 set by defining a permutation matrix

$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \tag{34}$$

and the diagonal matrix of the complex eigenvalues of $\mathbf{P}$

$$\mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & \sigma^2 \end{bmatrix} \tag{35}$$

the set of unitary trace-wise orthogonal matrices are obtained by [21]

$$\mathbf{U}_{ab} = \mathbf{P}^a \mathbf{Q}^b \ : a, b \in \{0, 1, ..., n-1\} \tag{36}$$

Written explicitly, the code group obtained from the Schwinger basis (36) is

$$\begin{aligned} \mathcal{G}_3 = \pm \Bigg\{ & \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \\ & \begin{bmatrix} 1 & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & \sigma^2 \end{bmatrix}, \begin{bmatrix} 0 & 0 & \sigma^2 \\ 1 & 0 & 0 \\ 0 & \sigma & 0 \end{bmatrix}, \begin{bmatrix} 0 & \sigma & 0 \\ 0 & 0 & \sigma^2 \\ 1 & 0 & 0 \end{bmatrix}, \\ & \begin{bmatrix} 1 & 0 & 0 \\ 0 & \sigma^2 & 0 \\ 0 & 0 & \sigma \end{bmatrix}, \begin{bmatrix} 0 & 0 & \sigma \\ 1 & 0 & 0 \\ 0 & \sigma^2 & 0 \end{bmatrix}, \begin{bmatrix} 0 & \sigma^2 & 0 \\ 0 & 0 & \sigma \\ 1 & 0 & 0 \end{bmatrix} \Bigg\} \end{aligned} \tag{37}$$

The transmission is encoded with (29), setting the initial transmission matrix to $\mathbb{H}_3$. One could also choose the initial matrix to be $\mathbf{I}_3$, but this choice reduces the resulting modulation to an antenna switching scheme. In the resulting differentially encoded transmission, only one antenna would be active at a given time slot, since in for all codes in $\mathcal{G}$, each row and column have exactly one non-zero element. Now $|\mathcal{G}_3| = 18$, so each code z_t can represent a 4-bit codeword. Two unique codes remain unused, which could potentially be utilized for synchronization purposes.

The rate of this code, again from (22) is $\log_2 16/3 = 4/3$. The resulting differentially encoded transmission forms a hexagonal constellation, where the possible symbol phase angles are separated by $\pi/6$. The symbols are at $\pm e^{2\pi kj/3}, k = 0, 1, 2$. Since the block code spans three transmission periods, ambiguity arises in choosing the right moment to form the received block from the received stream of symbols. There are two wrong choices between each correct time slot. Currently this ambiguity is solved by decoding all three choices $\{t,\, t+T_s,\, t+2T_s\}$ and correlating the received data against the unique preamble sequence. Finally, the procedure for $n = 4$ unitary basis construction is similar, starting with a 4×4 permutation matrix.

2.5 Direction of arrival estimation

Direction of arrival estimation will be briefly described, because it is almost an unavoidable demonstration for a coherent sensor array. The structure of the array is depicted in figure 5 for a uniform linear array (ULA). The N receiver antennas are equispaced, separated by distance d.

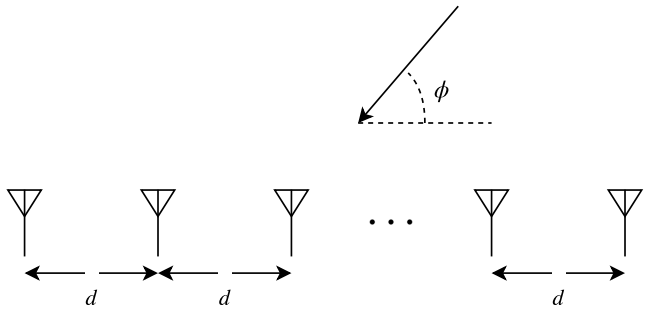

Figure 5: Signal impinging on an array of receivers

The signal is assumed to be a plane wave, originating from a point source at unknown direction at azimuthal angle ϕ. The data model for the incoming signal is

$$\mathbf{x} = \boldsymbol{\alpha}\mathbf{s}(\phi) + \mathbf{w} \tag{38}$$

where $\mathbf{s}(\phi)$ is the array steering vector, $\mathbf{w}$ a white Gaussian noise vector and $\boldsymbol{\alpha}$ the unknown amplitude. Choosing the steering vector indices such that the phase reference is at the center, the steering vector $\mathbf{s}$ can be written as

$$\mathbf{s}(\phi) = \left[z^{-(N-1)/2}, z^{-(N-3)/2}, ..., z^{-(N-3)/2}, z^{-(N-1)/2}\right]^{\mathrm{T}} \tag{39}$$

where $z = e^{jkd\cos\phi}$ and the angular wavenumber $k = 2\pi/\lambda$ [22]. A simple estimate for the direction ϕ is found by correlation

$$P_{corr}(\phi) = \mathbf{s}^*(\phi)\mathbf{x} \tag{40}$$

For the equispaced ULA array, the steering vector corresponds to Fourier coefficients and the DOA estimation reduces to the DFT of $\mathbf{x}$. The MUSIC method yields a better estimate, asymptotically unbiased and approaching the Cramer-Rao lower bound (CRLB). It does so by decomposing the signal into noise and signal subspaces and exploiting the known number of sources to exclude the noise [15, 22]. If it is known that there are M signals impinging on the array of N elements, MUSIC estimates their directions as a function of ϕ as

$$P_{music}(\phi) = \frac{1}{\mathbf{s}^\dagger(\phi)\mathbf{Q}_n\mathbf{Q}_n^\dagger\mathbf{s}(\phi)} \tag{41}$$

where the matrix $\mathbf{Q}_n$ spanning the noise subspace is found by eigendecomposition of the input covariance matrix $\mathbf{R} = \mathrm{E}[\mathbf{x}\mathbf{x}^\dagger]$. The idea is to partition the eigenvector matrix $\mathbf{Q}$ to signal and noise subspaces $\mathbf{Q}_n$ and $\mathbf{Q}_s$, corresponding to the respective eigenvalues, such that

$$\mathbf{R} = \mathbf{Q}[\boldsymbol{\Lambda} + \sigma^2\mathbf{I}]\mathbf{Q}^\dagger = \mathbf{Q}_s\boldsymbol{\Lambda}_s\mathbf{Q}_s^\dagger + \mathbf{Q}_n\boldsymbol{\Lambda}_n\mathbf{Q}_n^\dagger \tag{42}$$

Ideally the smallest eigenvalues in (42) are all equal to σ^2, but in reality they differ in magnitude. Also, in practice, the input covariance matrix must be estimated by

averaging over several input vectors. Then, $\mathbf{Q_n}$ are the eigenvectors which correspond to the $(N - M)$ smallest eigenvalues. The noise eigenvectors are orthogonal to the steering vectors $\mathbf{s}(\phi)$, therefore the inner product in the denominator (41) evaluate to zero, causing a peak in $P_{music}(\phi)$. [15]

The estimate for the signal directions $\hat{\phi}$ are the M highest peaks of $P_{music}(\phi)$. The lower variance of the method comes at a price of evaluating (41) for all possible directions ϕ, which is computationally demanding. [15, 22] The CRLB will not be derived here, but the minimum variance that any DOA method can reach is [22]

$$\mathrm{var}(\hat{\phi}) \geq \frac{6\sigma^2}{|\alpha|^2 N(N^2-1)(kd)^2 \sin^2\phi} \tag{43}$$

Clearly, the variance is dependent on SNR, the number of sensors and actual signal direction. The directions close to the normal of the antenna line, i.e. broadside, are the most accurate.

Ordinary MUSIC tends to fail in situations where there are closely spaced coherent sources. Constrained MUSIC exploits knowledge of the known signal directions to include them in the signal space $\mathbf{Q}_s$, decreasing the dimensions of $\mathbf{Q}_n$. This reduces variance for the true unknown direction estimates [23].

3 Research material and methods

This section describes the radio hardware and computational methods employed. The structure of the RTL-SDR receiver is introduced along with the FL2K VGA adapter. Additionally, the operation and limitations of the FL2K as a DDS RF transmitter is discussed.

3.1 RTL-SDR

The RTL-SDR, originally designed as a low-cost DVB-T tuner, later discovered by SDR enthusiasts, when it was found out that the device allows access to the raw IQ sample stream. Realtek, the company that designed the RTL2832U chipset, had intentionally implemented this functionality in the hardware, because their own proprietary driver provided software FM-demodulation for listening to radio broadcasts. That discovery took place in 2010, and by reverse-engineering the Linux driver released by the manufacturer, Steve Markgraf from Osmocom [7] developed the *librtlsdr* driver. The available portion of the spectrum is not the full 8 MHz DVB-T band, as most of the DVB-T decoding functionality resides in hardware. While theoretically the maximum sampling rate should be 3.2 MHz, the highest achievable stable sampling rate for the receiver has been found out to be 2.56 MHz [7].

A generic RTL-SDR USB dongle, with the plastic casing removed, looks like Figure 6. These receivers can be purchased from online marketplaces such as *Amazon* or *eBay*. The price of a generic dongle, not optimized for SDR usage is around 10 €[24, p. 5]. The specific device in Figure 6 was purchased from *eBay* for 6.5 €, including shipping.

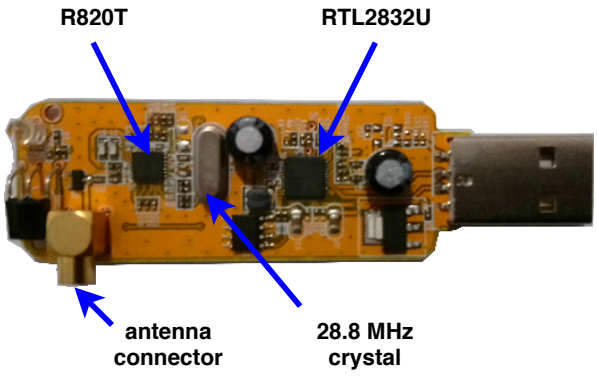

Figure 6: RTL-SDR dongle.

The device follows the ubiquitous superheterodyne principle, which means that the RF input signal is first converted to an intermediate frequency, after which the signal is passed to a quadrature demodulator. These two functions are performed by separate integrated circuits. The demodulator for compatible dongles is always the RTL2832U, but generic DVB-T dongle may contain a tuner from various manufacturers [7]. The features and performance of tuners from different manufacturers vary, this work

focuses only on hardware with the nearly identical Rafael Micro R820T and R820T2 tuners [7]. The tuner, depicted in the simplified block diagram of Figure 7, outputs an analog IF signal.

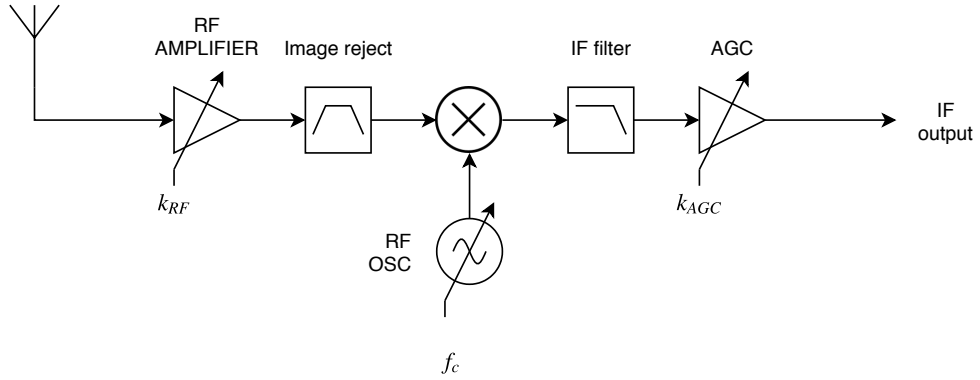

Figure 7: Rafael Micro R820T2 , adapted from [24, p.13].

The tunable parameters in Figure 7, tuning frequency f_c and RF gain k_{RF} are adjustable from software. The tuning frequency is generated from the 28.8 MHz crystal oscillator with a frequency multiplying PLL. Automatic gain control can only be enabled or disabled via software, the gain k_{AGC} is regulated by a control signal from the demodulator. All the control signaling between the two chips is carried on a I2C bus. [24]

The tuning range of the R820T/2 tuner is continuous, ranging from 24 MHz to 1766 MHz. The RF gain k_{RF} can be adjusted in 29 steps, from 0 to 50 dB [7]. The tuner output, at intermediate frequency $f_{if} = 3.6$ MHz, is then passed to the R2832U demodulator, depicted in Figure 8. Realtek has not released any documentation describing the chip, so most of the information comes from reverse-engineering original driver software. [24]

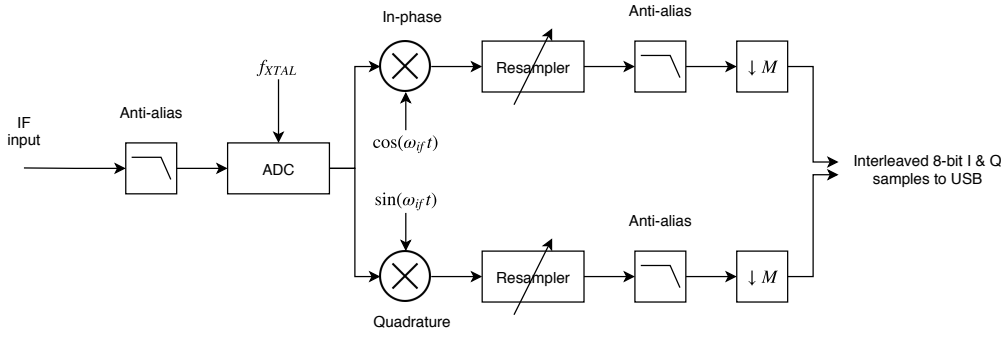

Figure 8: Realtek R2832U , adapted from [24, p.13].

An analog-to-digital converter operating at $f_{XTAL} = 28.8$ MHz samples the signal, after which the quadrature carrier mixing is performed with two NCOs. After

this, the signal is resampled with a fractional resampler, followed by a conventional decimation operation [24]. The fractional rate change serves also as a coarse carrier offset compensation. The receiver tuning offset can be considerable, and arises mainly from the low quality oscillator on the device [24]. By resampling the stream, but still interpreting it as if it were sampled with the original rate, the frequencies contained will naturally shift.

3.2 FL2K VGA-adapter

Transmitting RF signals intentionally from VGA display adapters is not a recent idea. Unintentional, information compromising RF emanation from displays, or more broadly computers, have been known and recognized by military and intelligence organizations since the 1960s [25]. In [25], Kuhn has shown how the image displayed on a monitor can even be reconstructed remotely by an eavesdropper from the received emanations. A method to intentionally transmit audio from a conventional analog monitor is also outlined in [25]. Naturally, the emanated signal power is weak, thus the receiver needs to be relatively close to the source [25]. In 2001, Erik Thiele published an open source tool, Tempest for Eliza, demonstrating broadcasting AM radio signal from a conventional VGA monitor [14]. The word Tempest was actually the code word for the classified US government program researching emission security [25].

What is relatively new is the discovery of uninterrupted streaming of samples from a VGA display adapter [14]. In 2018, Steven Markgraf from Osmocom [14], released driver software which opened up this possibility on inexpensive USB to VGA adapters. In almost all display adapters, the VGA color signal voltage is interrupted during the synchronizing horizontal and vertical blanking pulses, indicating the end of a single scanline and end of the display frame, respectively. Most modulation methods do not tolerate such interruptions in the signal. The Fresco Logic FL2000 enables the blanking signals to be disabled from software, permitting a continuous stream of samples. Essentially, the device is transformed into an inexpensive high speed multichannel digital-to-analog converter [14].

The 8-bit RGB color channel intensity values are written to a framebuffer, from where the VGA RAMDAC converts them to analog voltage. The VGA standard specifies the maximum voltage to be 0.7 V. That is, for each intensity value, 0 translates to 0 V and the 8-bit maximum 255 to 0.7 V. The screen resolution determines the sampling rate needed from the DAC, e.g. a 1280×1024 pixel resolution at a screen refresh rate of 60 Hz would translate to a $1280 \times 1024 \times 60\ Hz \approx 124$ MHz sample rate. The maximum sampling rate with the FL2K is reported to be 157 MHz, but this depends on the host computer USB3.0 chipset [14].

Outputting sample data through the USB bus at such high rates is computationally demanding, even more so when all three color channels are used. Computing the samples in real time would be possible only if the signal processing code was written in a low level language such as C or C++. For this reason, offline calculation of sample data was preferred, writing the samples into a binary file from Matlab. The *osmo-fl2k* source code [14] provides a tool for this. The tool was modified to enable

the output for green and blue color channels and to read interleaved 8-bit samples.

The output power spectrum of the FL2K VGA adapter was measured with the Rohde & Schwarz FSEA spectrum analyzer. For this purpose, the device was connected through a 1 nF capacitor to the analyzer input, forming a f_c =3 MHz highpass filter with the 50 Ω input impedance of the analyzer. A 10 MHz sinusoid was generated at a sampling rate of 100 MHz, representing the waveform with 10 samples per cycle. The power spectrum is depicted in Figure 9. The spectrum was also measured without a signal, for reference purposes.

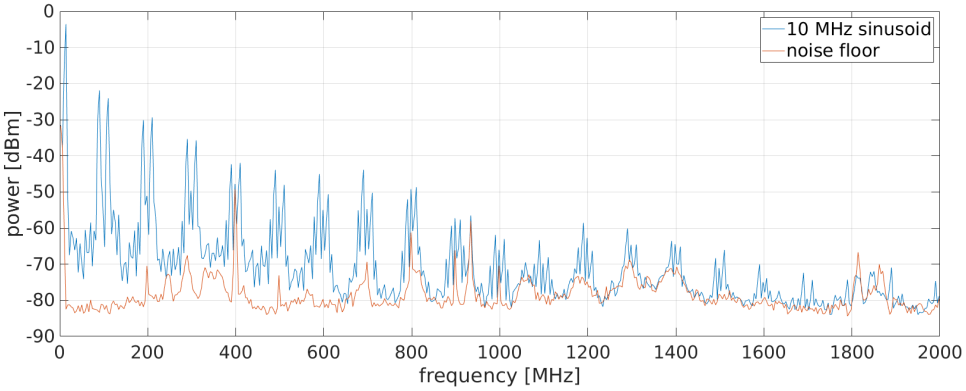

Figure 9: Time averaged power spectrum of 10 MHz sinusoid at 100 MHz sample rate.

The aliasing components extend beyond 1 GHz when the device is driving a 50 Ω load. Most of the peaks in the orange reference trace were verified to be leaking from the FL2K adapter itself, notably at 400 MHz. The reference peaking at 934 MHz occurs due to outside RF signals, because the coupling capacitor was unshielded and acted as an antenna. The frequency components fold at frequencies $f_N = |f_c - Nf_s|$, $N \in \mathbb{Z}$, as expected. Table 1 lists the power of the aliasing components at decade resolution. The values reported are the average of the two images around each integer multiple of f_s.

Table 1: Averaged power of $\pm N$ aliased frequency component at Nf_s

$f\ [MHz]$	10	100	1000	1500
$P\ [dBm]$	-3.5	-23	-62	-72

The peak-to-peak voltage from a VGA adapter is typically 0.7 V. This was verified with an oscilloscope. Without a load the voltage swing is $V_{pp} \approx 1.2$ V. When terminated with a 50 Ω resistance, the peak-to-peak voltage is $V_{pp} \approx 0.7$ V. It was expected the total power in the measurement setup would be approximately $P_t = \frac{v_{rms}^2}{R} = \frac{(0.7\ \text{V}/(2\sqrt{2})^2}{50\ \Omega} \approx 1.2$ mW. Measured total power was informally estimated by summing the frequency bins from the measurement of Fig. 9, as $P_t = \sum 10^{P_{dBM}/10}$. The result of this approximate measured total power is $P_t = 0.48$ mW.

When utilizing the FL2K as a transmitter, connecting the output pins from the RGB color channels directly to suitable antennas, the transmitting power is very limited. Especially so, when operated in the range above 1 GHz, where the antenna dimensions of the MIMO system are practicable. Still, an imitated GPS signal, which operates in frequency slightly above 1.5 GHz, transmitted with the FL2K, has been successfully received. Albeit the receiver was extremely close in that demonstration [14].

Considering the usage of FL2K to transmit modulated signals as such, everything up to passband output needs to be done in software. Essentially, this is a form of Direct digital synthesis (DDS). That is contrary to commercial SDRs, that expect user to input complex baseband samples, which are then modulated to a user defined carrier frequency. The steps needed for digital modulation, transforming samples from input symbol s to passband signal x are depicted in Figure 10.

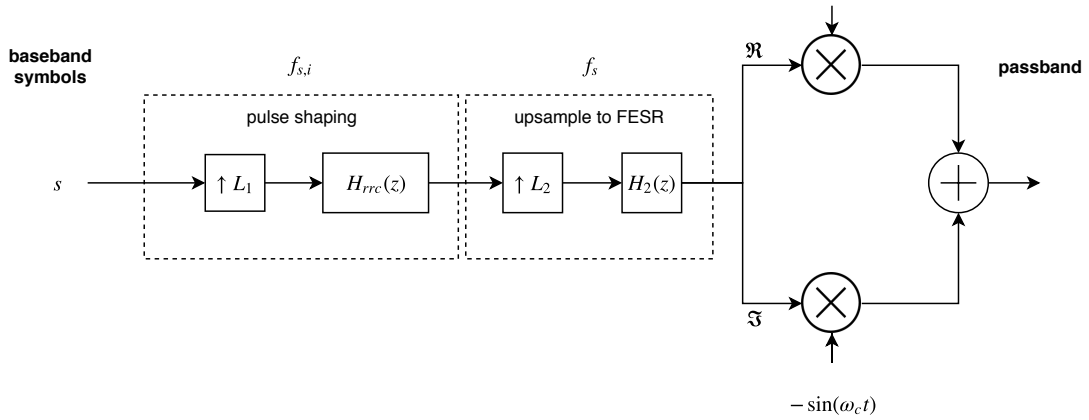

Figure 10: Block diagram of transmitting software.

In Fig. 10, a root raised cosine baseband pulse shaping filter $H_{rrc}(z)$ limits the signal bandwidth, removing imaging after upsampling to intermediate rate $f_{s,i}$. Another upsampling is needed to reach frontend sample rate f_s, the rate at which FL2K operates. Quadrature modulation is performed by mixing the signal with In-phase and Quadrature carriers. The resulting samples are converted to signed 8-bit integer format and written to a binary file. For three transmit antennas, the operations are performed thrice, writing the transmission file in an interleaved format. In the experiments, the front end sample rate was limited to $f_s = 100$ MHz. The offline transmission file grows quickly, at 100 MHz one second translates to a 300 megabyte file. For continuous transmission, the file can be repeated from start, but care should be taken that the transition from end to start is glitch free.

The use of RF spectrum is regulated across the world. It could certainly be illegal to operate a transmitter amplifying an unfiltered signal from the FL2K, given the range of the aliasing harmonics. Even if the intended broadcasting frequency was license free, the aliasing components might collide with a regulated band. Without amplification though, the emitted power is negligible and causing harmful interference

is therefore unlikely.

Finally, one can easily envision the device used as a starting piece in building amateur SDRs by adding filtering, mixing and amplification. Since the DACs it contains have such a wide bandwidth, an intermediate frequency and quadrature modulation could be performed entirely in software, as in Figure 10, leaving only upconversion and image rejection to be implemented with analog electronics.

3.3 Computational methods

This section describes some of the relevant signal processing methods needed in the receiver software implementation and communication experiments. The software depends heavily on cross-correlation, which would not be feasible to implement in time domain. Some method of symbol timing synchronization was necessary when evaluating the performance of the space-time codes. Additionally, the discrete root-raised cosine filter used in the symbol timing error detector is also defined here.

3.3.1 Cross-correlation

The unnormalized estimate for cross-correlation of two complex sampled sequences $\mathbf{g}$ and $\mathbf{h}$ of equal length N can be written

$$\hat{\phi}_{\mathbf{gh}}(l) = \sum_{k=l}^{N-1} g[k]h^*[k-l] \quad 1 \leq l \leq N-1 \tag{44}$$

where l is the displacement, or lag. Scaling the resulting $\hat{\phi}$ with $\frac{1}{N-|l|}$ or $\frac{1}{N}$ yields the unbiased and biased estimate for correlation, respectively [26, p. 853-854].

Cross-correlation finds broad range of usage in signal processing, when a measure of similarity between two data sets is needed. For instance, cross-correlation can be employed in finding a known preamble sequence from a received signal that is contaminated with noise. Notating the cross-correlation function as elements of a vector

$$\boldsymbol{\phi}_l = \hat{\phi}_{\mathbf{gh}}(l) \tag{45}$$

in the range $-N \leq l \leq N$. The complexity of computing the vector $\boldsymbol{\phi}$ trivially with (45) is $O(N^2)$. Computing the same vector in transform domain with the help of FFT, the complexity reduces to $O(N \log N)$. The result is numerically equivalent, disregarding the negligible floating-point error [26, p. 853-854].

The correlation is closely related to the convolution operation, which in turn can be efficiently performed in transform domain [26, p. 853-854]. Comparing the discrete convolution operation

$$(g * h)[n] = \sum_{k=-\infty}^{\infty} g[k]h[n-k] \tag{46}$$

with (44) underlines the similarity. Convolution in time domain is equivalent to multiplication in transform domain. To be precise, the equivalence is actually

referring to the circular convolution [27, p. 123-125]. The elements of vector $\boldsymbol{\phi}$ are members of the DFT pair [28, p. 648]

$$\boldsymbol{\phi}_l \Longleftrightarrow G_k H_k^* \tag{47}$$

where G and H refer to the DFT of g and h, respectively. The procedure to compute the cross-correlation efficiently via FFT is:

- append both vectors **g** and **h** with $N-1$ zeros
- compute $\boldsymbol{G}$ and $\boldsymbol{H}$ with FFT of length $2N$
- conjugate multiplication in transform domain, $\boldsymbol{\Phi}_k = \boldsymbol{G}_k \boldsymbol{H}_k^\dagger$
- inverse FFT, $\boldsymbol{\phi} = \text{IFFT}(\boldsymbol{\Phi})$

There are variations to this procedure. Instead of complex conjugation, one could time-reverse one of the inputs. The resulting vector will have its component indices in a wraparound order, where the positive lags from $0, 1, ..., N-1$ correspond to elements $\boldsymbol{\phi}_0, \boldsymbol{\phi}_1 ..., \boldsymbol{\phi}_{N-1}$ and the negative lags mirrored from $-1, -2, ..., 1-N$ correspond to $\boldsymbol{\phi}_{2N-1}, ..., \boldsymbol{\phi}_N$ [28, p. 648]. A simple solution to restore the indice wrap-around back to continuous order exploits the circular nature of the convolution by prepending one of the inputs with N zeros, instead of appending.

3.3.2 Discrete root-raised cosine filters

Discrete root raised cosine $h_{rrc}[n]$ is obtained by the impulse invariance method, i.e. sampling the response of (15) by $h_{rrc}(nT)$. Because $h_{rrc}(t)$ has infinite support, the sampled response must be truncated to a finite integer, L, number samples on both sides of $t = 0$. The resulting discrete filter spans the interval $-LT_s \leq t \leq LT_s$, approximating the response with $2L + 1$ samples. Smaller values of roll-off β require larger L, since at small excess bandwidth $h_{rrc}(t)$ decays longer. This can be seen from the simplified plot in Figure 11, where the effect of adjusting roll-off β is visualized in both time- and frequency domain. The bandwidth occupied by a root raised cosine pulse is simply $B = (1 + \beta)/T_s$ [16, pp. 686-686].

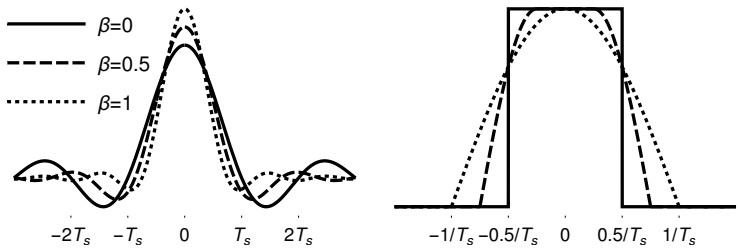

Figure 11: On the left side, pulse shape $h_{rrc}(t)$ and on the right side $|H_{rrc}(f)|$, while $\beta = 0, 0.5, 1$.

Below, a piecewise definition for calculating the coefficients of a discrete root raised cosine filter is given, derived by sampling the continuous response. There are other formulations of (48), which may yield differing gain. By representing the number of symbols per sample by $N = T_s/T$ and scaling the gain by sampling time T, a discrete root raised cosine is given by the piecewise function

$$h_{rrc}[n] = \begin{cases} \frac{1}{\sqrt{N}}\left[1+\beta\left(\frac{4}{\pi}-1\right)\right] & n = 0 \\ \frac{\beta}{\sqrt{N}\pi}\left[\pi\sin\left(\frac{(1-\beta)\pi}{4\beta}\right) - 2\cos\left(\frac{\pi(1+\beta)}{4\beta}\right)\right] & n = \pm\frac{N}{4\beta} \\ \frac{1}{\sqrt{N}}\frac{\sin\left(\frac{n\pi(1-\beta)}{N}\right)+\frac{4\beta n}{N}\cos\left(\frac{n\pi(1+\beta)}{N}\right)}{\frac{\pi n}{N}\left[1-\left(\frac{4\beta n}{N}\right)^2\right]} & \text{otherwise} \end{cases} \tag{48}$$

where $n = -L, 1-L, .., L-1, L$. The resulting filter coefficients are symmetric w.r.t center $n = 0$, meaning the filter is linear phase. Truncating the response necessarily gives rise to frequency domain sidelobes, the level of which depend on both L and β. The truncated filter will not fully comply with the no-ISI condition (12) of the ideal continuous raised cosine. ISI can still be minimized to negligible levels with reasonably low filter lengths, even with filters spanning only $L = 3$ symbols on both sides of current symbol [16, pp. 686-686].

The standard procedure for pulse shaping between transmitter and receiver goes as follows. The symbols to be transmitted, sampled at $1/T_s$ are first upsampled by the factor N to the rate $1/T$. The resulting sequence is convolved with $h_{rrc}[n]$. At the receiver, an identical $h_{rrc}[n]$ is applied before downsampling to symbol rate. Here it is assumed that the receiver samples at the correct moment. Effectively, the signal is now convolved with the raised cosine shape. The factor N, interpreted as symbols per sample is an integer value. Generally, values $N \geq 2$ are used. [16, pp. 686-686]

3.3.3 Symbol timing synchronization

Symbol timing synchronization is the task of estimating the optimal sampling instant from the received signal. The data clock is unavailable to the receiver in a wireless system. There exist feed-forward and feedback based method, of which the former is more popular MIMO applications. A feed-forward synchronizer extracts the timing from training symbols in the transmission [16, p. 514]. The symbol timing synchronizer used in the experiments is a feedback structure, and can be called a symbol timing PLL. It consists of a loop filter, a timing error detector (TED) and a NCO. [16, pp. 434-437]

The signal model at the receiver is

$$y(t) = G_a \sum_k a_k h_{mf}(t - kT_s - \tau) + w(t) \tag{49}$$

where G_a represents the total amplitude gain and τ is the unknown timing delay. The approach in Figure 12 used in this work is based on the maximum likelihood timing error detector (MLTED). MLTED estimates the timing error with the product of the matched filter output and the output of a derivative matched filter. Since the signal is complex, the implemented TED in Figure 12 sums the outputs of in-phase

and quadrature MLTEDs [16, p. 497]. As a whole, the symbol timing PLL works to drive the timing phase error $\tau_e = \tau - \hat{\tau}$ to minimum [16, p. 479].

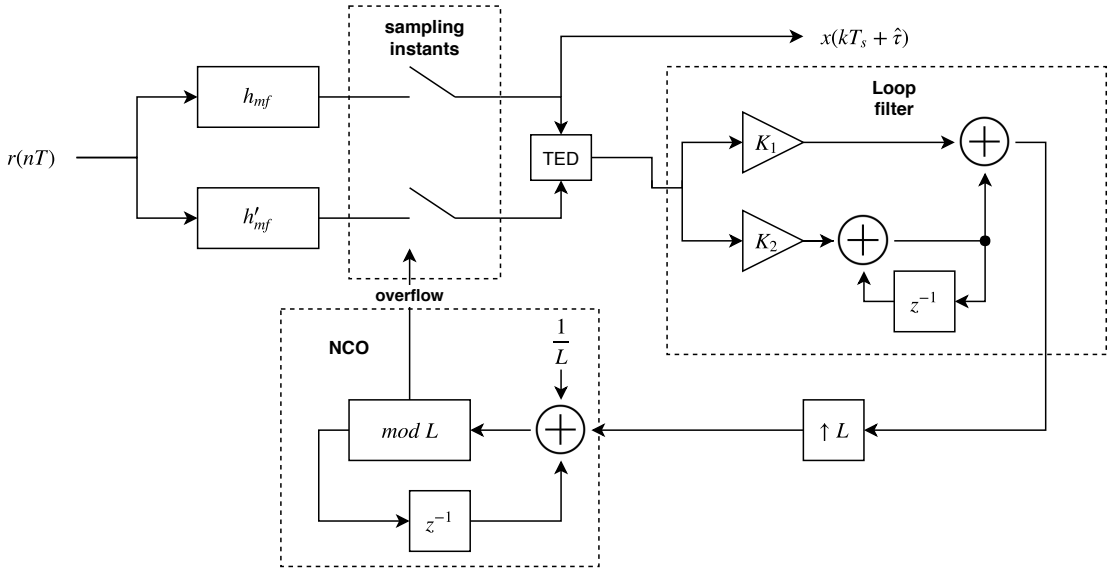

Figure 12: Symbol timing synchronization with ML TED, adapted from [16, p. 479].

For the receiver array, the structure was modified such that the signals are first combined by equal-gain combining. Then the sampling instants were extracted to sample the individual signals, with the assumption that the physical length of the antenna array is much smaller than symbol length.

4 Receiver design

This section describes the design of the hardware and software of the proposed SDR receiver. Circuit design for the common clock and reference signals are discussed. A prototype receiver was assembled and revised with the concepts described here. The software Section reviews the structure of the program and discusses the techniques applied in synchronization and phase correction of samples.

4.1 Hardware

To the best of our knowledge, the first successful attempts at achieving phase coherence on multiple RTL-SDR were made by Juha Vierinen, while working on a low-cost passive radar [8]. In his implementation, one of the receivers had the crystal removed. The oscillator input inverter pin of that slave dongle was connected via capacitive coupling to the master dongle. Once the receivers were clocked from a common source, their streams were synchronized to a common reference signal. This approach suffices for an array of few dongles, but the oscillator of the master dongle cannot supply the current to drive the input capacitance resulting from a larger array of parallel slaves without buffering.

Tatu Peltola's design connected more dongles to a common clock, by buffering the clock output from the master dongle with 74 series logic inverters [9]. In his work, a common reference noise signal was distributed via inductive coupling to the signal dongles, while the master dongle received only noise. Peltola had also designed a printed circuit board (PCB) for the noise couplers. The same PCB is utilized in this work.

The design proposed here continues along the same direction, with a separate clock and reference generator. The concept is depicted in Figure (13).

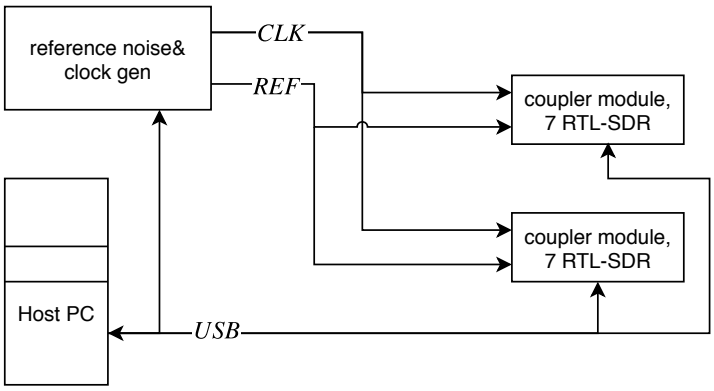

Figure 13: Coherent receiver with two coupler modules

Each coupler module incorporates a 7-port USB hub to accommodate the RTL-SDR dongles. A photograph of a coupler is shown in Figure 14. On the left side of the image, next to the BNC antenna connectors, are the antenna input strips

and neighbouring noise strips. This structure forms directional couplers between the strips. That is, the reference noise is distributed via a transmission line coupling effect, i.e. crosstalk. The reference noise and clock signals are brought on the circuitboard from separate BNC connectors on the right side of the PCB, not visible in Fig. 14.

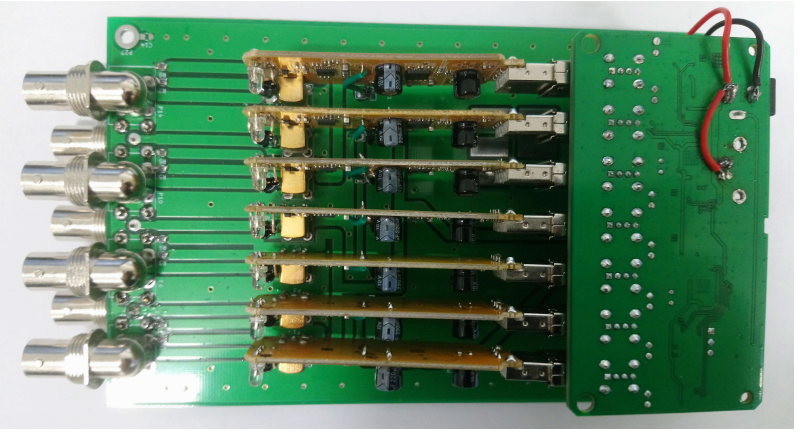

Figure 14: Coupler module

The material costs of a single coupler module are around 130-150 €. If the PCB is ordered from a cheap prototype manufacturer in quantities of ten, the price per unit is approximately 7 €. The powered USB hub can cost 30 €, depending on the manufacturer. We have used D-Link and Targus HUBs. The RTL-SDR will cost approximately 7×10 €. The most expensive components on the PCB are the SMA and BNC connectors, of which there are 16. Again, if these are ordered online from the Far East, the price per unit is approximately 1.5 €.

The master dongle is housed inside the clock generator unit, receiving only the reference noise. The clock- and reference signal circuits are discussed in the following sections.

4.1.1 Clock generation

In the system designed here, the clock generation is entirely separated from the dongles. All dongles will have the crystal removed, and are instead clocked by a separate Pierce-Gate topology oscillator, for which the circuit diagram is shown in Figure 15.

In the circuit, inverter U_1 shifts the phase ideally by $-180°$ relative to the input. The reactive components, C_1, C_2 and X provide the remaining $-180°$ phase shift to satisfy the Barkhausen criterion for sustained oscillation. That is, the closed-loop gain around the circuit $k \geq 1$, and the phase shift around the loop is an integer multiple of 2π. Initially, capacitances C_1 and C_2 can be assumed equal and tweaked afterwards for tuning the oscillator, if need be. The feedback resistor R_f functions to linearize the gain of the inverter, similarly to negative feedback in an OP-AMP circuit. The series resistor R adjusts the current through the feedback, isolating the output of the inverter from the reactive network. [10]

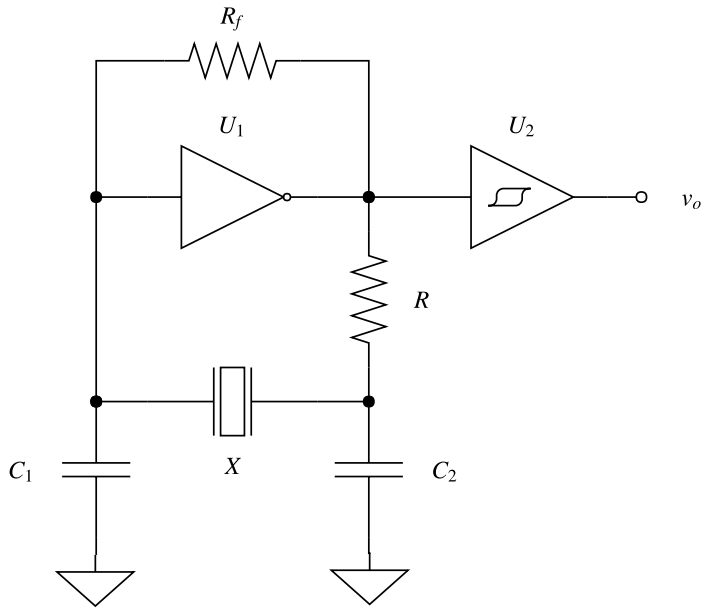

Figure 15: Pierce-Gate oscillator circuit

The Schmitt trigger inverter U_2 is included to generate a more ideal square wave and guard against spurious triggering. Following the design procedure described in [10], the capacitance seen by the parallel resonant crystal should ideally equal the internal load capacitance of the resonator

$$C_{load} = \frac{(C_{in} + C_1)(C_{out} + C2)}{C_{in} + C_{out} + C_1 + C_2} + C_{stray} \tag{50}$$

where C_{in} and C_{out} are the input and output capacitances of the inverter driving the oscillation. C_{stray} is an estimated value of paracitic capacitances arising from circuit board strips [10].

Since the system is to be built from inexpensive and easy-to-source parts, a crystal desoldered from one of the dongles was used. The manufacturer is not known, no datasheet is available, therefore a typical value of $C_{load} = 20$ pF was estimated for the load capacitance. This choice was supported by Farnell catalog, in which most crystals in the relevant frequency range have a load capacitance between 20 and 30 pF.

The commonly available Texas Instruments SN74HC04 was chosen for inverter U_1. This discrete IC provides six independent inverters: the remaining 5 could be used for buffering the outputs. The Schmitt trigger is from the same series, 74HC14 with 6 independent inverters. SN74HC04 datasheet states a value of $C_{in} = C_{out} = 10$ pF [39]. Estimating the board stray capacitances to be $C_{stray} = 5$ pF, which is a bit high since the prototype construction will be non-ideal. Since $C_1 = C_2 = C$ and $C_{in} = C_{out} = C_l$, solving (50) for C reduces simply to

$$C = 2(C_{load} - C_{stray}) - C_l \tag{51}$$

Inserting the values discussed above, $C_1 = C_2 = 20$ pF. The closest readily available value, 22 pF, was used in the prototype. Series resistor R can then be solved for the RC circuit formed by R and C_2 [10]

$$R = \frac{1}{2\pi f C_2} \tag{52}$$

Inserting values, $R \approx 251.2\ \Omega$ at the desired frequency of $f = 28.8$ MHz. The closest value available in R12 resistor series is conveniently close, 250Ω. Being a first order RC network, it will impose a $-\pi/4$ phase shift on the signal at the oscillation frequency before the crystal.

In the final version of the clock generator prototype, the output v_o in the circuit of Figure 15 is still buffered with 74HC04 inverters, each inverter driving a module of 7 RTL-SDR dongles. The rack enclosure in which the prototype was built provides 9 BNC connector outputs of the equiphasic clock signal.

4.1.2 Reference noise properties

The common reference signal used in synchronization should have low temporal autocorrelation. For an ideal white noise process $W(t)$ the autocorrelation is the Dirac delta function [29, pp. 93-99]

$$R_W(\tau) = \sigma^2 \delta(\tau) \tag{53}$$

Some authors omit the variance term in (53), due to the properties of the Dirac delta. In a discrete time system with sampled white noise $W[n]$

$$R_W[n] = \sigma^2 \delta[n] \tag{54}$$

the infinite Dirac delta is replaced by the Kronecker delta $\delta[n]$ [29, pp. 93-99]. Equation (54) simply implies, that the magnitude of the autocorrelation peak is proportional to the noise power, denoted here by the variance σ^2.

From the viewpoint of synchronization, the reference noise is now rather unusually the desired signal, that is corrupted by regular noise and even information carrying radio signals received by the antennas. Only the reference receiver, which does not connect to an antenna, mostly excludes external RF signals. Still, all the receivers will capture noise caused by the circuitry itself, e.g. thermal noise and shot-noise. It is reasonable to assume that this type of electronic noise is uncorrelated among the receivers. During synchronization, undesired correlated interference enters the sampled stream from exterior sources.

One problematic source of correlated interference is the 28.8 MHz clock signal, which has been recorded leaking into the received stream even in the unmodified RTL-SDR. Testing on a vanilla dongle shows, that for instance the 34th harmonic of the clock signal, $35 f_{clk} = 1008$ MHz, is visually observed in the received spectrum as a peak aproximately 25 dB above the noise floor. Because it is a square wave, the fundamental frequency and its odd harmonics are present well in the higher end of the tuning range.

Leakage and coupling are well-known challenges in mixed signal designs, circuits containing both analog and digital sections. In the design of the noise source, clock leakage must be carefully considered, because the noise generator contains a high gain block. In the worst case, clock signal leakage through RF into the noise preamplifier is subject to the same overall gain as the desired signal. Also, it was noticed that the USB NRZI signaling into the received stream. The bitrate of highspeed USB 2.0 is 480 kbps, and at 480 MHz, a peak of approximately 10 dB was observed.

Initial tests were made with digital pseudorandom noise as the reference signal, connecting the FL2K VGA adapter output to the system. This worked up to around 1.5 GHz, but the setup was impractical to operate and tied the resources of one extra computer. A microcontroller with sufficiently high GPIO speeds could perhaps be utilized for the reference signal, with suitable custom output circuitry. However, an on-chip DAC that many microcontrollers have is useless in this task, since these usually contain a reconstruction filter limiting the bandwidth. The FL2K tests mentioned relied on the unattenuated aliasing above the sampling rate. In such setup, it should also be ensured that the generated pseudorandom sequence is long enough. A frequently repeating sequence might cause the synchronization algorithm to detect a wrong cycle and synchronize to a mistaken instant. The benefit of knowing the synchronizing sequence would also remove the need for a separate reference dongle.

The decision was made to generate the wideband noise with analog circuitry. Vierinen [8] had noted that retuning the receivers did not affect the relative timing of the sample streams, which would be a desirable property. To determine whether this claim holds, and does it hold with a massive array of receivers, a wideband reference signal was preferred. Reliable reference signal was also important during developing the software.

4.1.3 Zener diode noise generator

The Zener diode is known to produce noise, when reverse biased. That is, the zener breakdown voltage has been exceeded and current flows from cathode to anode. The application of Zener diodes in noise generators was first proposed by Susans [13] in 1967, with a circuit design covering frequency range 30-900 MHz. Susans also determined the magnitude of the noise to be related to Zener current. In 1975 Somlo [12] presented measurements of the noise characteristics w.r.t Zener voltage, concluding that the excess noise increases with larger Zener voltages. In their recent paper [11], Arslan and Yıldırım propose a design for a broadband noise source utilizing two Zener diodes, biased at different operating points to flatten the output spectrum. In [11], Zener diodes with breakdown voltages between 15V and 24V were tested, finding the best performance with the 22V diode. They remark that the 22V diode was from a different manufacturer than the rest of the lot, and that this may affect the result [11]. Figure 16 depicts the operating principle of a Zener diode noise generator circuit.

The voltage v_{cc} must be above the Zener voltage of the diode below for the breakdown to occur. The constant current source, which can be adjustable to match the diode properties, keeps the current through the diode in an optimal region.

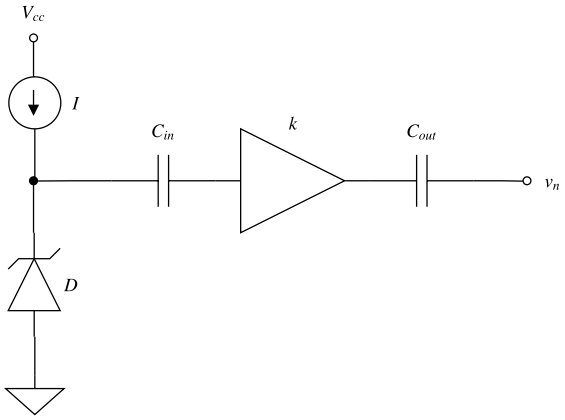

Figure 16: Zener diode noise generator model

Capacitive coupling C_{in} prevents the DC biasing voltage from amplifier input, C_{out} blocks the potential DC offset at the amplifier output stage.

In the prototype design, based on [30], a 24V Zener diode produces the weak noise voltage which is then amplified by a chain of three MMIC broadband amplifiers. The amplifiers proposed in [30] were replaced with products from NXP semiconductor, since the original parts manufactured by Mini-Circuits could not be sourced. The amplifier chain starts with BGA2867, followed by another BGA2867 and ending in BGM1013. An inverting switching regulator, Texas Instruments TL497A is utilized for generating a 34V voltage from the 12V operating voltage for the Zener diode. The 78L05 regulator is configured as an adjustable current regulator, to tune the Zener drive current. The RC correcting networks for flattening the output spectrum [30] were simplified to capacitive coupling, since in this application the spectrum does not need to be optimally flat, nor the probability distribution exactly Gaussian. What matters most is noise power above the thermal noise floor. Complete diagram of the circuit is given in Appendix A.

Additionally, the output stage amplifiers were designed to be enabled and disabled from software. After the synchronization is achieved, the reference noise becomes useless and only degrades the SNR of the signal dongles. This was accomplished by switching off the output amplifier operating voltage. An inexpensive USB-connected microcontroller, STM32 was specifically programmed for this purpose. It registers as a serial device, in Linux filesystem the path of the device is typically */dev/ttyACM0*.

The progam running on the microcontroller simply pulls a GPIO pin to high logic level when certain character is sent from the host. That GPIO pin connected to a load switch constructed from a discrete PMOS-NMOS FET transistor pair, to achieve high side switching. Low side switching, i.e. switching the circuit path to ground could be accomplished with a single NMOS. High side switching was chosen since positive power rail is more likely to contain higher levels of interference, such as leaked clock signal. Low side switching would also lift the ground potential for the output stage slightly, due to the NMOS drain-source resistance.

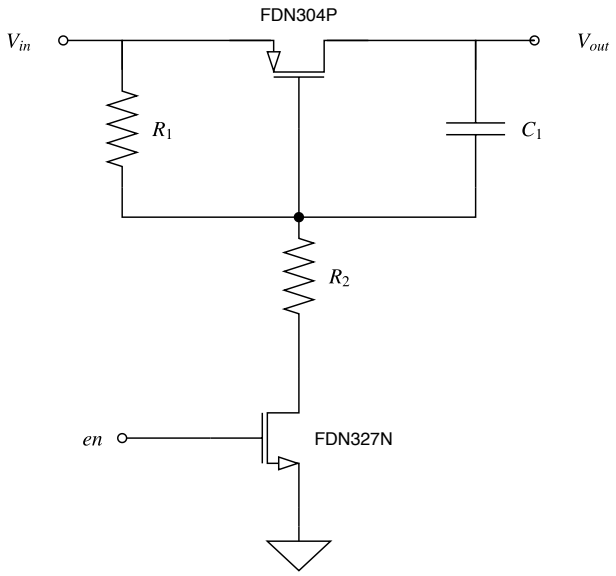

Figure 17: High side load switch for switching off the reference noise.

The circuit 17 switches the $V_{in} = 12$ V operating voltage directly and is intended to be followed by a 5V regulator. A low voltage drop L4941BV regulator with 1A current specification was used in the prototype. Signal *en* is the active high 3.3V logic level control from the microcontroller. Capacitor C_1 functions as a primitive inrush current limiter to protect the gate of the FDN304P PMOS. Voltage divider between resistors $R_1 = 10$ kΩ and $R_2 = 8.2$ kΩ limits the gate-source voltage to $R_1/(R_1 + R2)12$ V ≈ 5.5 V, omitting the drain-source resistance, because its value is in the mΩ range and thus negligible. Limiting the gate-source voltage is necessary, because the FDN304P datasheet [31] specifies an absolute maximum of $V_{gs} = \pm 8$ V. Switching before regulation was chosen to avoid a voltage drop due to IR losses from FET drain-source resistance. It would also allow in-place regulation, i.e. bringing the voltage conversion closer to the load.

In the assembled device, there are 9 BGM1013 output stages. Their input signal is split from the output of the middle amplifier in the chain. The complex impedance of the coupler module is not known, although it could be measured with a vector network analyzer. Each coupler module terminates the directional coupling strips with 8 parallel 51 Ω resistances. If there is a resonance in the system, where the complex impedance cancels out, the noise output stage sees a strictly real impedance of 50/8 Ω = 6.25 Ω. To avoid overloading or even damaging the output stage, the load impedance should not be decreased by connecting more couplers in parallel. For this reason the device includes separate output stages for each coupler module.

When the clock and noise generators were first assembled in a 1U rack-mount casing, the device worked, but it was observed with an oscilloscope that the clock indeed leaked into the noise output ports. To reduce radiated coupling, the case was partitioned with a grounded dividing plate. To reduce coupling via power supply conductors, power supply filtering circuit of Figure 18 was added to the sensitive

analog side. These modifications suppressed the leaking considerably, although the clock leakage is still observable, if not in the time domain view of an oscilloscope, but in a spectrum analyzer.

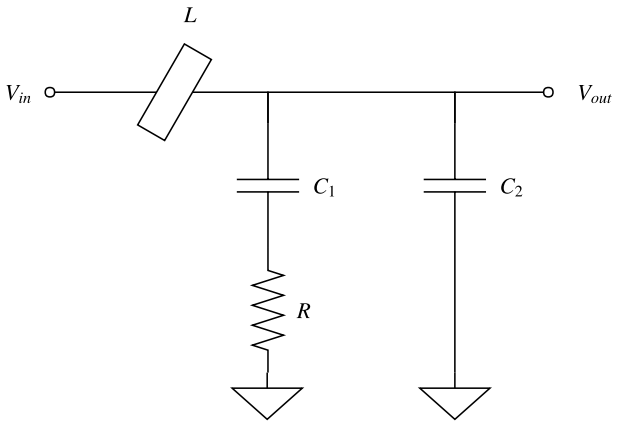

Figure 18: Power supply filter for the noise generator.

In Figure 18, the component values are $C_1 = 1$ µF, $R = 10\ \Omega$ and $C_2 = 100$ nF. L is a ferrite bead, with reasonably large AC resistance. The purpose of the RC_1 network is to damp the possible resonant peaking arising from LC_2.

The generated noise was measured with a Rohde & Schwarz FSEA spectrum analyzer. Measurements were conducted with the output stage enabled and disabled. Additionally, a reference measurement was included where the device was unpowered, to estimate the noise floor. Clock coupling is still evident from the peaks in the spectrum traces of Figure 19, when output stages are enabled. The slope of the noise exhibits roughly a first order lowpass 20 dB per decade roll-off. Around 1 GHz, the noise appears to leak to the output even when the output stages are switched off.

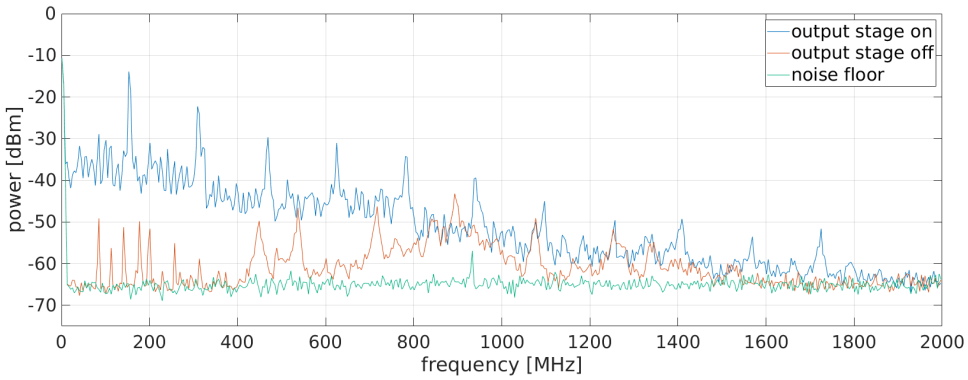

Figure 19: Time averaged noise power spectrum in frequency range 0 to 2 GHz.

Despite the imperfections, synchronizing was still attained up to approximately 1.5 GHz. To revise the flaws present in the prototype, a few possible solutions could be tried. The noise- and clock generators could be placed in physically separate

enclosings, or at least implement a design with split ground planes for the analog and digital sections. To reduce the attenuation towards higher frequencies, one could try to alter the Zener voltage and current. A proper circuit board could also help to lessen the effect of parasitic capacitances and inductances.

4.2 Software

Software for the coherent receiver was written in C++, linking against several accelerated libraries. It compiles into a command-line application, which streams the received samples through the ZMQ messaging protocol. The sample data can then be read by another application or a remote computer on the network. The ZMQ [32] messaging libraries are free software, available for a variety of programming languages. Here, it is utilized as an extension built on top of TCP/IP, providing easier interfacing and greater flexibility. The main advantages over TCP/IP are arbitrary message length and simplicity of interfacing. Contrary to TCP/IP behaviour, a ZMQ socket publishing messages do not require a client to be connected before messages can be sent. The handling of client connections is abstracted from the programmer, performed automatically by the ZMQ library. [32]

The software links against an experimental fork of *librtlsdr* [33]. This modified driver library allows disabling dithering, which was noted to destroy phase coherence. It has been noticed by several developers, that if dithering is enabled, the relative phases of the samples will drift.

When the software starts, it identifies the connected receivers by serial number and initializes asynchronous read threads and control threads for all requested devices. An optional configuration file can be specified at startup, defining the serial numbers of the receivers to be initialized. The configuration file may also be used to specify the order of received streams in the receive matrix.

As it takes considerable time to initialize a dongle, approximately one second per receiver, the asynchronous read threads halt execution at a barrier object until all the threads have completed the initialization. After all the threads are initialized, the program enters the main loop where the lag values are calculated via cross-correlation, samples are converted from unsigned to signed and published to the ZMQ socket. The control threads alter the sample rate of the signal dongles asynchronously according to the lag values, until time synchronization is reached. The sequence diagram of the receiver software is shown in Figure 20. For the purposes of this diagram n refers to the total number of connected receivers, including the reference dongle.

The control threads were deemed a necessary addition, because it proved to be impossible to control the sample rate from the process context of the *librtlsdr* callback. Had it not been so, this callback function could have provided a relatively accurate time reference with resolution $blocksize \cdot T$ seconds. This was actually tested by recording timestamps from the system high resolution clock inside the callback.

The cross-correlation is computed in the frequency domain, as discussed in Section 3.3. For efficient implementation, the computation of FFT and IFFT rely on the library *"Fastest Fourier Transform in the West"* FFTW. FFTW is a project originating from MIT, freely available under the GNU General Public License. The

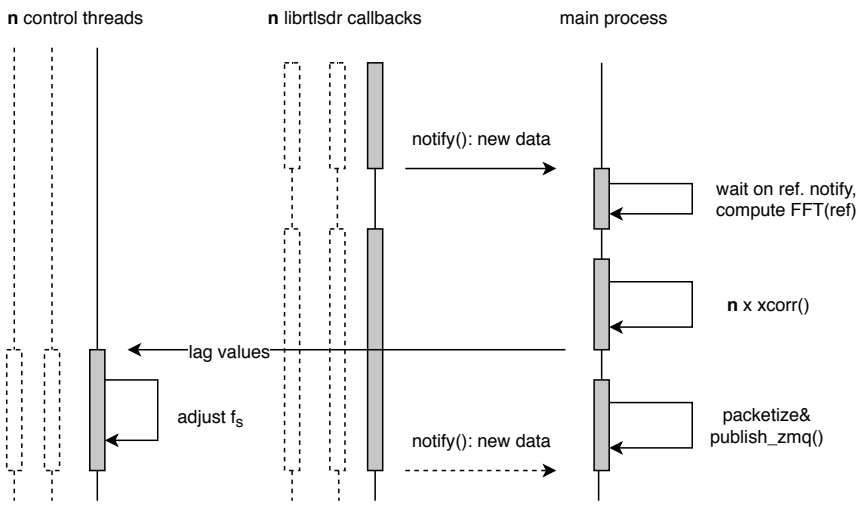

Figure 20: Sequence diagram of the coherent receiver software.

project claims performance comparable to vendor-tuned libraries [34]. The FFT is calculated with single precision 32-bit floating point accuracy.

Vector-Optimized Library of Kernels (VOLK), a sub-project spawned from GNU radio, was chosen to accelerate operations on vectors. This library offers common mathematical functions operating on vector parameters, manually tuned to take advantage of CPU vector instructions, such as SSE on Intel platforms [35]. In the software, it is used for the element-wise frequency bin multiplication in cross-correlation and sample conversions between 8-bit integer and 32-bit floating point data.

The USB specification defines 7-bit addressing scheme, where the address 0 is reserved. Therefore, the maximum number of endpoints is 127. In addition, each 7 port USB-hub consumes one unique address [36]. Theoretically, if there were no other devices connected, the maximum number of receivers on one host is 111. That means 15 full hubs and one with six dongles. This limit concerns hosts operating in the enhanced host controller interface (EHCI) mode, meaning USB2.0. For USB3.0, the extensible host controller interface (XHCI) type hosts the maximum is much lower and depends on the chipset. In our case, XHCI maximum was 24. It is unlikely though, that neither the software as such nor the hardware could handle that much traffic.

Finally, to receive the data for processing in Matlab, Zeromq-matlab originally written by Stephen McGill [37] was modified to read the protocol header and directly form the received matrix. While this could have been done in Matlab, it would have been cumbersome and slower.

4.2.1 Network protocol

A single ZMQ message contains a $L \times N$ matrix of complex samples. Each message starts with a header, described in Figure 21, which is always $12 + 4N$ bytes long.

The number of channels N includes the reference noise, which is always located at received channel 0. Reference noise was included for testing purposes, such that the correlation peak could be verified from samples in Matlab.

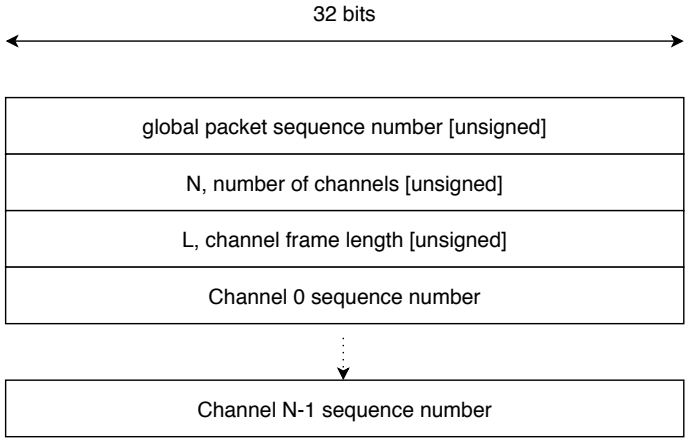

Figure 21: Sample data protocol header.

By sending the dimensions of the matrix in every message, it would be possible to change the number of channels without restarting the software. The rest of the message is the actual sample data, represented by interleaved signed 8-bit I and Q components. The length of the payload is thus $2NL$ bytes, where L is the constant block size received from the devices. By default, $L = 16384$ samples, but the value can be specified when invoking the software.

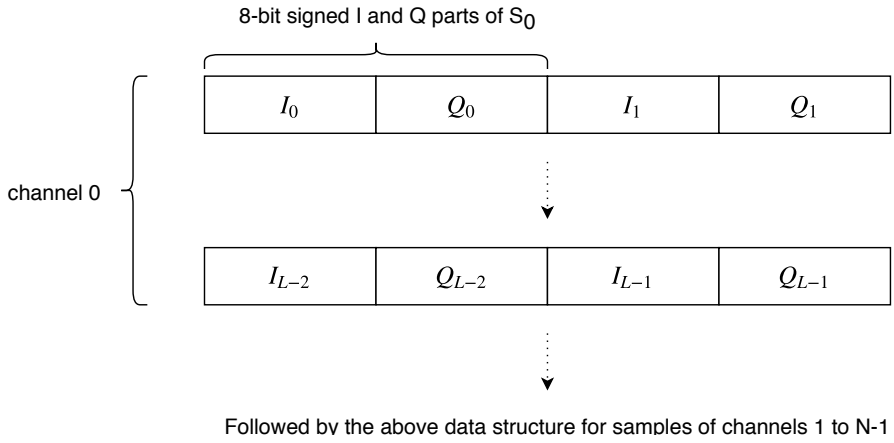

Figure 22: Sample data format.

Matlab is suited to rapid testing of DSP algorithms, but suffers from poor performance in real-time applications. Generally, receiving matrices with $N > 10$ channels will clog the Matlab side and cause buffering to grow. Samples are not dropped however, the ZMQ buffer expands to accommodate the data not processed

in time. The samples published to a ZMQ socket can also be read by other software, it is not restricted to Matlab. Functionality was tested with GNU radio ZMQ source block, but here the protocol header is naturally not supported and had to be removed. It would also be trivial to write a tool to store the received data on disk.

4.2.2 Fractional sample index timing

After the correlation function is evaluated, the location of the maxima is sought from the absolute value of the complex correlation $|\hat{\phi}_{\mathbf{xr}}(l)|$ of signal vector $\mathbf{x}$ and reference $\mathbf{r}$ from the master dongle. One such correlation is plotted in Figure 23. This maximum may often fall between sampling instants, suggesting sub-sample differences in timing, perhaps due to clock phase imperfections of the system. Examples of this phenomenon are pronounced in the zoomed plots of Figure 24. Ideally, the correlations on both side of the peak would be of equal magnitude. On the left side of Fig. 24, it appears that the true maximum is between instants 3 and 4. Hence, interpolation is employed to find the fractional timing difference.

Many techniques exist for realizing a fractional delay. Most of these result in a FIR structure, with the exception of allpass filter based IIR designs. Sophisticated FIR methods, which include for instance the windowed sinc function and multirate polyphase methods, yield more accurate results at the cost of increased complexity [38]. Lagrange polynomial interpolation falls into the FIR category and was chosen for the implementation because of its simplicity. Also, since finding the fractional sample index of the cross-correlation peak is actually an inverse interpolation problem, the recursive allpass methods are not applicable.

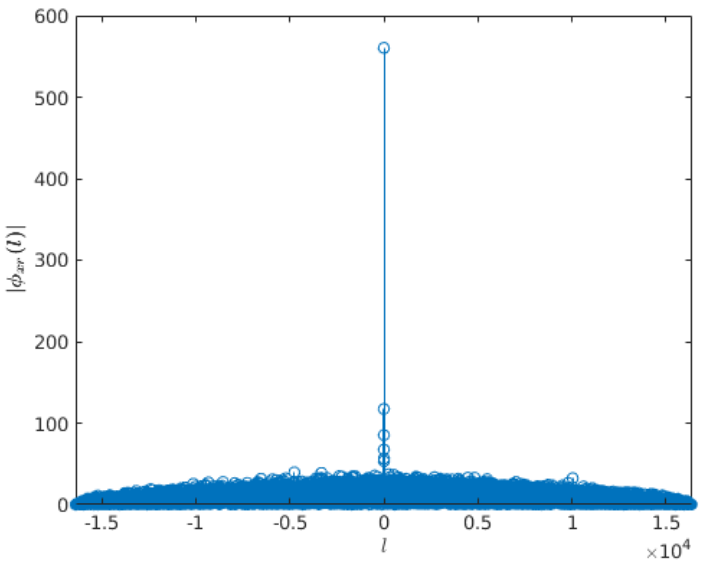

Figure 23: Cross-correlation against the reference channel, linear magnitude scale.

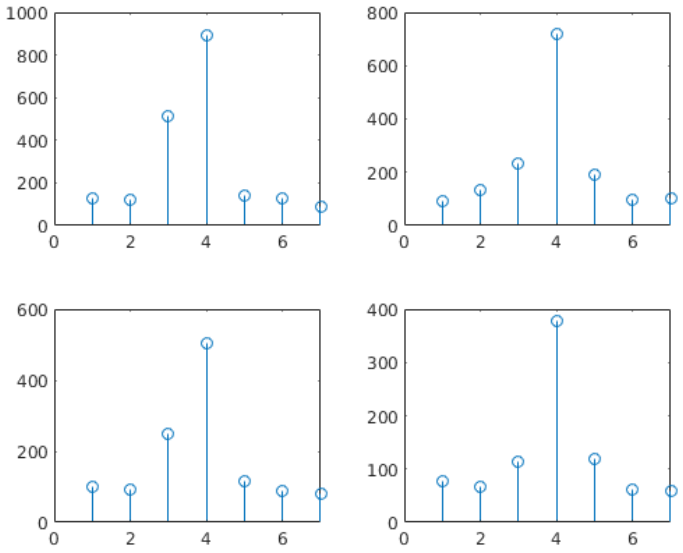

Figure 24: Correlation against the reference channel zoomed on the peak at index 4.

The $n+1$ point Lagrange interpolation polynomial is defined by [40]

$$L(x) = \sum_{k=0}^{n} y_k \ell_k(x) \tag{55}$$

where the basis functions are

$$\ell_k(x) = \prod_{j=0, j\neq k}^{n} \frac{x - x_j}{x_k - x_j} \tag{56}$$

The derivative interpolating polynomial can be calculated from

$$L'(x) = \sum_{k=0}^{n} y_k \ell'_k(x) \tag{57}$$

where the differentiated basis functions are

$$\ell'_k(x) = \sum_{i=0, i\neq k}^{n} \left[\frac{1}{x_k - x_i} \prod_{j=0, j\neq (i,k)}^{k} \frac{x - x_j}{x_k - x_j} \right] \tag{58}$$

Let the cross-correlation peak be located at sample index m. If n is even, there would be an unequal number of data points neighboring m. Selecting odd n avoids the balancing choice by centering the interpolation on m. The resulting polynomial from $n+1$ point interpolation with (55) is of order n, but the derivative (57) decreases the polynomial order to $n-1$ [40, p.75]. Reasonable choices to interpolation range are 5 and 3 points, yielding third and first degree polynomials. Adding more points would require numerical methods to find the roots. Interpolating with three points centered at m, at indices $(m-1, m, m+1)$ corresponding to the absolute values of

the correlation function $(|\hat{\phi}(m-1)|, |\hat{\phi}(m)|, |\hat{\phi}(m+1)|)$. Notating $|\hat{\phi}(m)| = s_m$ and the fractional sample index with d. Reordering terms from (57) and (58) we have

$$a = -s_{m-1} - 2s_m + s_{m+1}$$
$$b = -\frac{1}{2}s_{m-1} + \frac{1}{2}s_{m+1}$$

$$L'(d) = ad + b \tag{59}$$

The result is a linear equation and the location of the peak is solved trivially, by setting the derivative L' to zero

$$d = -\frac{b}{a} = \frac{s_{m-1} - s_{m+1}}{2(s_{m-1} - 2s_m + s_{m+1})} \tag{60}$$

For 5 point interpolation, again centered on m, in matrix form to outline the similarity to Farrow filter matrices. The input vector is now $\mathbf{s} = [s_{m-2}\ s_{m-1}\ s_m\ s_{m+1}\ s_{m+2}]^T$ and the derivative is approximated by

$$\mathbf{G} = \begin{bmatrix} \frac{1}{6} & -\frac{2}{3} & 1 & -\frac{2}{3} & \frac{1}{6} \\ -\frac{1}{4} & \frac{1}{2} & 0 & -\frac{1}{2} & \frac{1}{4} \\ -\frac{1}{12} & \frac{4}{3} & -\frac{5}{2} & \frac{4}{3} & -\frac{1}{12} \\ \frac{1}{12} & -\frac{2}{3} & 0 & +\frac{2}{3} & -\frac{1}{12} \end{bmatrix}$$

$$L'(d) = \mathbf{dGs} \tag{61}$$

where $\mathbf{d} = [d^3\ d^2\ d\ 1]$. The resulting equation is now a 3rd degree polynomial. The coefficients of powers of d in this equation are the inner products: $a = d^3\langle \mathbf{g}_1, \mathbf{s}\rangle$, $b = d^2\langle \mathbf{g}_2, \mathbf{s}\rangle$, $c = d\langle \mathbf{g}_3, \mathbf{s}\rangle$ and $e = \langle \mathbf{g}_4, \mathbf{s}\rangle$, where $\mathbf{g}_r$ denotes the rows of matrix $\mathbf{G}$. A well-known closed-form solution for the roots of a cubic equation exists and is described in the following paragraph.

First, converting the polynomial to monic form $d^3 + b'd^2 + c'd + e'$, by dividing $b' = b/a$, $c' = c/a$ and $e' = e/a$ [40, p.23]. Then, canceling the second order term, reducing to form $w^3 + pw + q$, by setting $d = w - 1/3b'$. Evaluating the discriminant with

$$D = -4p^3 - 27q^2$$

where $p = (3c' - b'^2)/3$ and $q = (2b'^3 - 9b'c' + 27e')/27$. Then computing

$$A = \sqrt[3]{-\frac{27}{2}q + \frac{3}{2}\sqrt{-3D}}$$
$$B = -3p/A$$

The three roots are obtained by

$$\begin{aligned} d_1 &= \frac{1}{3}(A+B) - \frac{1}{3}b' \\ d_2 &= \frac{1}{3}(\rho A + \rho^2 B) - \frac{1}{3}b' \\ d_3 &= \frac{1}{3}(\rho^2 A + \rho B) - \frac{1}{3}b' \end{aligned} \tag{62}$$

where $\rho = e^{2\pi j/3}$ [40, p.23]. The correct root in this case must be real, neglecting the floating point precision error in the imaginary part, and in the range $-1/2 \leq d \leq 1/2$. In the above method, the intermediate results are complex, even though the imaginary parts may cancel out in the result. This also means, that complex arithmetic needs to be used for the root finding in the software. The added computational burden is still light, relative to the computation of cross-correlations for all channels against the reference noise. The estimation of d would in any case be performed only every L samples, the block size received from RTL-SDR.

Another option for solving d is to use iterative numerical methods, like the Newton-Raphson's method. The simpler 3-point method (60) was however preferred in the implementation, as the more complicated alternative did not yield noticeable difference. When the reference noise is on and synchronization is reached, the fractional part d mostly varies only in the third decimal place.

4.2.3 Time synchronization

The *librtlsdr* driver already establishes a number of buffers to hold the received samples. To keep the software simple and avoid implementing more buffering, the synchronization is completely achieved by altering the sampling rate of the signal dongles. Peltola [9] had used the same approach to implement a fractional delay in his software. Here, both integer and fractional parts of the delay are compensated with the same method. The delay relative to the reference dongle, which is held as the constant mark the signal dongles are measured against, now consists of integer and fractional parts $l = m+d$.

Essentially this approach takes advantage of the hardware resampler of R2832U, which is normally used in correcting frequency offsets, discussed in Section 3.1. For the typical dongle, the offset falls in the range of ten to hundred ppm, negative or positive. The offset is compensated for by altering the sampling rate of the stream. This can be taken advantage of to advance or to delay the stream of samples, even to fractions of sample index. Altering the overall sample rate slightly to align the buffers was also tested during developement and proved unsuccessful.

The sampling rate and its reciprocal, sampling time, when corrected with ppm value are

$$f_s' = f_s + pf_s \Rightarrow T' = \frac{T}{1+p} \tag{63}$$

where f_s is the actual sample rate, f_s' the sample rate after correction and $p = ppm \cdot 10^{-6}$.

Consider two sample vectors $\mathbf{r}$ and $\mathbf{x}$, sampled with exact same sampling time. Suppose that $\mathbf{r}$ is the reference, which is sampled at the correct epoch whereas $\mathbf{x}$ is delayed by lag l. The aim is to drive the argument of maximum of the correlation function to zero. Having the ability to individually control the sampling rate of both vectors, the streams could be synchronized by accelerating or decelerating one of them until the correlation peaks at zero. Now considering the vectors contained only the ideal reference noise, but with arbitrary lag, such that $r[nT] = x[(n+l)T']$. Thus, equating the indices

$$T' = \frac{nT}{n+l} \Rightarrow f_s' = f_s \frac{n+l}{n} \tag{64}$$

As the achievable correction p is limited in range, the natural choice for the number of samples n is the block size, the number of samples in each received buffer, or an integer multiple thereof.

Practical considerations arising from lack of sufficiently accurate hardware timers also support this choice. Generally, in a non real-time system, a timer will only guarantee that it awakens the execution at some epoch after the requested period has elapsed, never before. It was noticed that a Linux interval timer can vary in the order of tens of milliseconds. There is no easy way to rely on system timers for accurate and deterministic control. If there was, the amount of cross-correlation computation could be drastically reduced, by solving the time needed from (63) and (64).

The software currently implements an ad hoc synchronization, which relies essentially on brute force, since the above could not be implemented in reasonable time. The cross-correlation function is evaluated for each channel every block until a synchronization achieved flag is raised. The corrected sample rate f_s' is decreased by piecewise steps when the lag reaches $|l| < \{500, 50, 10, 1\}$. After the last step, when synchronizing to fractional sample indices, the correction is set to the lowest possible value. This was necessary to avoid overshoot, since the rate is adjusted only every $blocksize \cdot T$ milliseconds. Perhaps a compromise between timers and brute force could be devised, where the coarse timing offset is handled with an interval timer.

As mentioned, there are limitations to the maximum ppm correction. The range of ppm determined from the RTL-SDR source code is $\pm 488 ppm$. The requested ppm correction is represented by a 32-bit integer, which is multiplied by $2^{24}/10^6 \approx 16.8$, cropped to 14 bits and written to the registers lower 8 bits first, upper 6 bits second. In other words, the RTL2832U resampler register accepts 14-bit values in the range [-8192,8191]. The resolution of writing into this register is also limited, because of the aforementioned multiplication. The experimental driver [33] providing a function without the unnecessary scaling was used to achieve finer control of the resampler.

4.2.4 Phase correction

After the streams are time-aligned, the relative phases of the streams still need to be corrected. The tuner PLL frequency synthesis introduces seemingly random phase rotations to the downconversion local oscillator. A unit magnitude phase correction coefficient α can be computed from the dot product of the reference noise vector $\mathbf{r}$

and signal $\mathbf{x}$ by

$$\alpha = \frac{\langle \mathbf{x}^*, \mathbf{r} \rangle}{|\langle \mathbf{x}, \mathbf{r}^* \rangle|} \tag{65}$$

for each signal channel. Obviously, this is equal to the zero-lag correlation, which is already computed. It was discovered that the phase correction does not remain constant after retuning. This would make sense, since the tuner PLL needs to reacquire lock when tuning to another frequency. Even when tuning back to a frequency where α was calculated after visiting another frequency, recalculation is needed. Phase-coherence is thus available only on frequencies where the reference signal has sufficient power. However, α appears to remain stable for relatively long periods.

Drifting of relative phase was observed on few occasions, after the software had been running for approximately half an hour. For extended periods of operation, it could be sufficient to estimate the phase correction factors when tuning the receiver and recalibrate periodically every 10 minutes, by switching the reference noise on briefly.

5 Results

In this section, the results are presented. First, the computational requirements of the software are evaluated. Then, the results from DOA estimation are shown. In the last part, the performance in MIMO DUSTM communication experiments is presented.

5.0.1 Software performance

The CPU usage of the software, measured with Linux command *top*, is reported in Table 2. The values were measured on a 3.3 GHz Intel 4-core Xeon PC, running Debian Linux. The percentage reported by *top* is relative to the number of CPU cores, for a quad core CPU, the maximum consumption is 400%. A 100% utilization means the resources of single core are exhausted. Values before synchronization is achieved, after synchronization with and without a subscriber are given. It is evident that the computational load reduces significantly after synchronization is achieved. Connecting a client to read the samples slightly increases the load. This happens because the ZMQ call to function *socket.send()* returns instantly when ZMQ determines there are no subscribers.

Table 2: CPU usage versus the number of signal channels n, pre & post synchronization and with a subscribed client. $f_s = 1$ MHz.

	CPU %			
n	**4**	**8**	**16**	**35**
not synch.	17	32	58	95
synch.	7	9	14	36
synch & client.	11	16	21	52

After the synchronization flag is enabled for a signal channel, the cross-correlation function for that channel is only evaluated for every 32nd block of samples. The blocks where the evaluation occurs are alternated, to balance the computational load. In its current structure, the lag computation is single-threaded and is not performed in time before new data arrives if CPU consumption exceeds 100%. Thus, reaching the synchronized state where the computational load decreases may start to fail if even more signal channels are added.

Compiling the software for the Raspberry Pi 3 is possible, but requires manual compilation and installation of the libraries *FFTW* and *VOLK*. The limited hardware was able to handle 4 signal channels at a 500 kHz sampling rate. Adding more signal channels or increasing the sampling rate overburdened the cross-correlation thread. By optimizing the synchronization, a Raspberry Pi might be able to control an array of one coupler module, or 7 dongles. Such a device could be used for a remote, network connected receiver array.

5.0.2 Phase and timing synchronization

Tests were conducted to find out whether the timing must be synchronized every time the receivers are tuned to a new frequency. Retuning remotely on the fly is possible with the *controlcmd* tool. Sampling frequency in these tests was 1 MHz. It was discovered, that the relative timing of the streams was indeed not affected by the operation and even the fractional part from (60) remained stable. This was tested with $n = 9, 18$ and 35 signal dongles on frequencies where the reference signal is optimal, producing a strong peak in the cross-correlation function.

This means that the receiver array can operate over the whole tuning range of the RTL-SDR simply by retuning from a stable reference frequency, and that the requirements for the reference signal bandwidth are reduced considerably. Phase correction using (65) is however unusable outside the stable frequencies.

The synchronization appears to remain steady for extended periods. The longest period the system has been running continuously is approximately 4 hours. When the subsample differences in relative timing are estimated with (60) and corrected for, the magnitudes neighboring the correlation peak are quite well balanced, as seen in Figure 25. The difference to Fig. 24 is obvious.

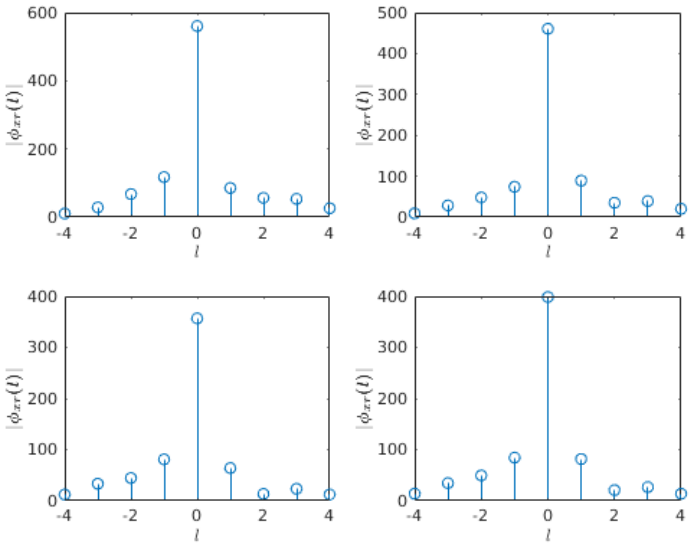

Figure 25: Correlation against reference channel with synchonization to fractional sample indices.

The relative phase spectrum measured from a few signal dongles, with reference noise enabled is plotted in Figure (26). In the plot, the relative phase is evaluated from the 2048 point DFT of signals $\mathbf{x}_1$ and $\mathbf{x}_2$ by multiplying the respective frequency bins, i.e. $\angle(X_1 \odot X_2)$. In this measurement, external signals were excluded by disconnecting antennas. It was discovered during these measurements, that while the timing of the streams remains stable after retuning, the phase correction factors do not. The LO frequency synthesis phase locked loop in the R820T tuner is the most likely the cause of this.

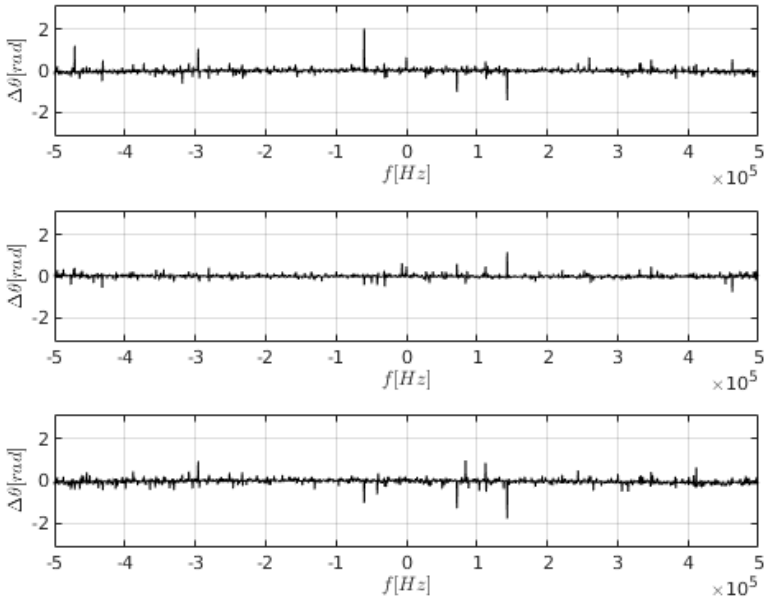

Figure 26: Phase spectrum difference, in one coupler module. $f_s = 1$ MHz at tuning frequency 1250 MHz.

Taking the signals from two dongles in different coupler modules and connecting their antenna inputs to an FL2K outputting pseudorandom noise via a splitter and adequate attenuation, the relative phase looks like Figure 27. The phase noise does increase, but the overall trace is centered on 0 rad.

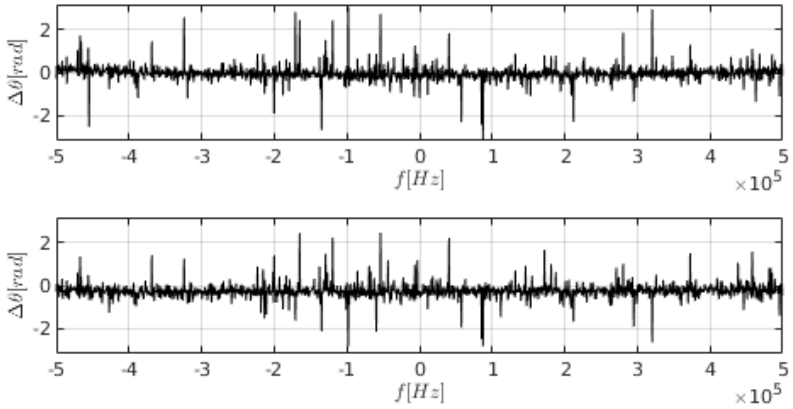

Figure 27: Relative phase across couplers, common input signal. $f_s = 1$ MHz at tuning frequency 1250 MHz.

5.0.3 Direction of arrival measurements

Direction of arrival was estimated with the MUSIC method outlined in Section 2.5. The distance of the signal emitter from the ULA sensor, consisting of simple ground plane antennas, was approximately 2 meters. The operating frequency was 1225 MHz. The $N = 9$ antennas are separated by distance 0.125 m, for which half-wavelength $\lambda/2$ corresponds to a 1200 MHz frequency.

The number of snapshots for averaging the covariance matrix was 32. What was puzzling at first was that even with no test signal source on, (41) exhibited a strong peak exactly from broadside direction. It became evident that the source of this peak was the reference noise. All the sensors receive it at the exact same phase, which interprets to a source location at exactly 90 degrees as seen in Figure 28. This was checked by disabling the noise, which affected the magnitude of the peak.

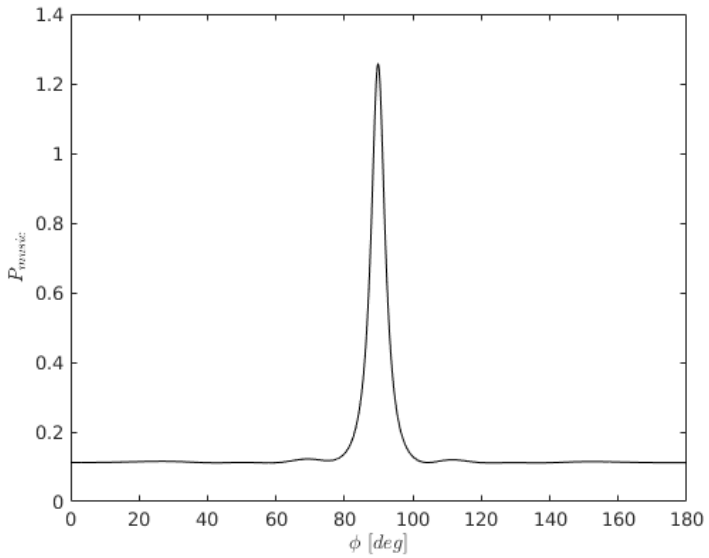

Figure 28: Plot of P_{music} with reference noise enabled.

Constrained MUSIC [23] could be used to exclude the effect of the reference noise, since the direction is known exactly. However, subsequent testing was performed by disconnecting the reference noise cables after synchronization was acquired. Figure 29 shows few plots of P_{music} when using the FL2K adapter to transmit a 1225 MHz signal and moving the antenna across the broadside of the array.

Again, the received signal power is very limited at this frequency, thus the relative magnitude of noise and signal eigenvalues is also comparatively small. This experiment did make it obvious that the automated switching of the reference signal performs poorly at these frequencies. This can be seen from the noise power measurement in Figure 19 in Section 4.1.2.

The FL2K outputs more power at the lower aliasing harmonics. Reducing the number of antenna array elements to 4 by picking only a section of the array and by tuning to 625 MHz, the peak P_{music} power increases, as seen in Figure 30. Naturally, the reduced number of elements $N = 4$ decreases the accuracy.

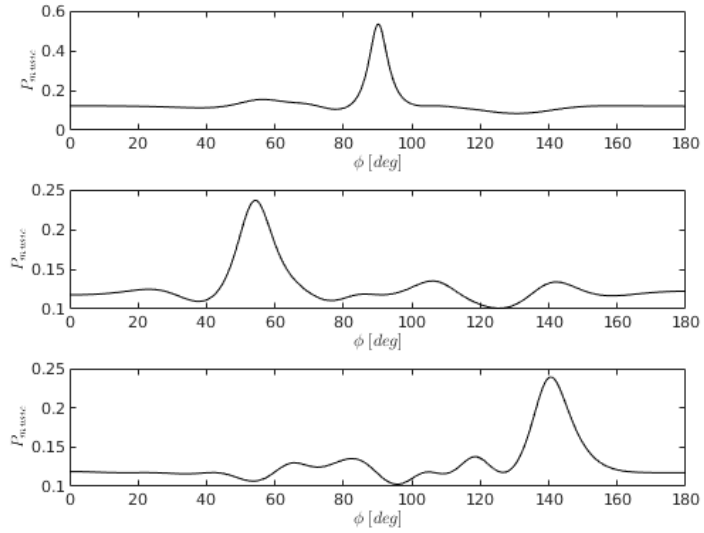

Figure 29: P_{music} at frequency 1225 MHz.

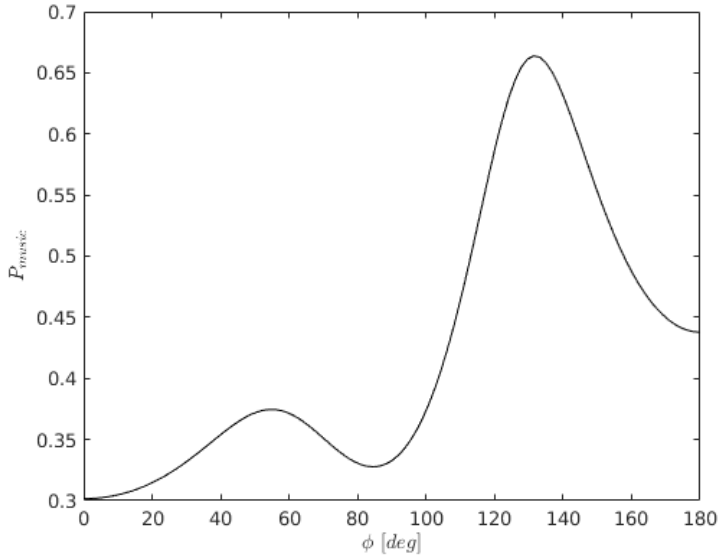

Figure 30: P_{music} at frequency 625 MHz.

5.0.4 Differential unitary space-time modulation experiments

These transmit-receive experiments were conducted at a frequency of 826 MHz. As before, the FL2K VGA adapter functions as the signal source. The transmitter antenna array is approximately at a 3-meter distance from the receiver array and a line of sight exists between them. Increasing the distance would lead to greater path losses and lower received SNR as the transmitter power cannot be increased. In the experiment setting, the received mean sample SNR is approximately 5 dB.

It should be mentioned that since the code is differential, strict phase-coherence is not required at the receiver. Stable phase offsets caused by the tuner downconversion LO will be canceled in the differential detector. They can be thought to be included in the term recognized as the channel estimate in (31).

At the transmitter, an offline transmission file was calculated. Two symbol rates were tested, $R_S = 62.5$ ksym/s and $R_S = 125$ ksym/s. For the 2×2 code, this yields bit rates 94 kbps and 188 kbps by $|\mathcal{G}|R_S/2$. At an intermediate sample rate of 1 MHz, these correspond to 16 and 8 samples per symbol, respectively. Pulse shaping was performed with a root-raised-cosine filter, which had a roll-off factor of 0.35. The received mean sample SNR drops about 3 dB when there are 8 samples per symbol compared to 16 samples per symbol.

The frames are identified at the receiver by a unique preamble. The preamble was a maximum length sequence of 30 bits for the 2×2 code and 32 bits for the 3×3 code. This way, the preamble bits are coded in blocks separate from information bits. A pseudorandom sequence of 300 bits was generated, grouped into 3 bit chunks and mapped to codes from codegroup $\mathcal{G}$ according to Definition (28). The resulting 2×2 blocks were encoded with (29), followed by upsampling and pulse shaping and again resampling to 100 MHz for transmission with the FL2K.

The figures 31 and 32 display the bit error rate w.r.t receiver antennas. These BER plots were produced from the same recorded data set and averaged over 100 frames. The bit error rates obtained with a nonsynchronous sampling receiver for the 2×2 code are shown in Figure 31. As expected, BER decreases when the number of receive antennas increases.

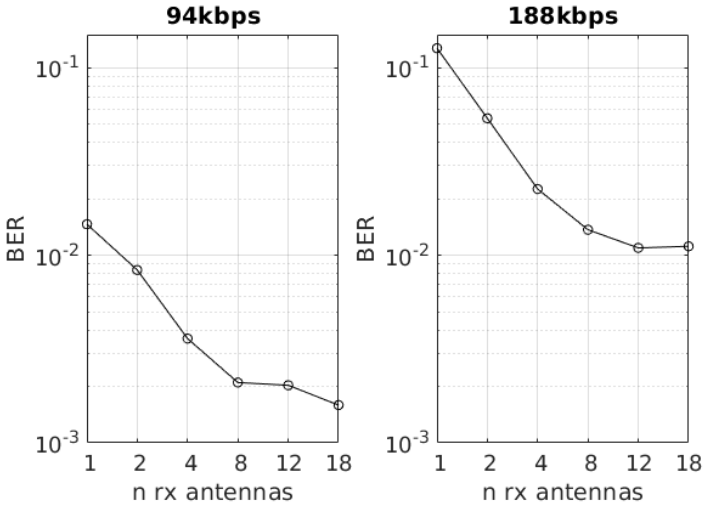

Figure 31: 2x2 DUSTM BER.

When the symbol timing PLL is enabled, synchronizing the sampling instants to the equal-gain combined sum signal, the BER performance improves, as seen in Figure 32. For the lower bit rate, BER drops to zero starting from 8 receive antennas.

The 3×3 DUSTM scheme is encoded similarly, by grouping the information bits in groups of 4 bits and mapping to codegroup from Definition (37). The block

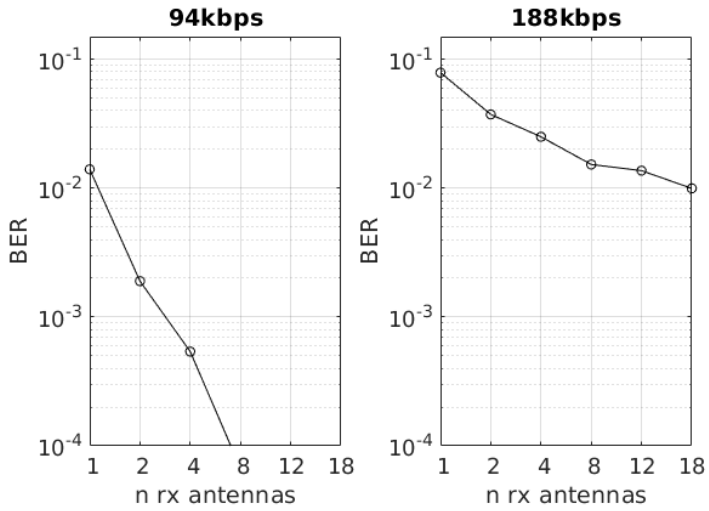

Figure 32: 2x2 DUSTM BER with symbol timing synchronization.

code now spans $p = 3$ transmission slots, thus the bit rates are $4R_S/p$, yielding 83 kbps and 166 kbps. In Figure 33, again, BER drops to zero between 4 and 8 receive antennas for the lower bit rate. It appears that the symbol timing PLL does not acquire a stable lock when there are only 8 samples per symbol. Hence, the results are affected by the quality of symbol timing synchronization.

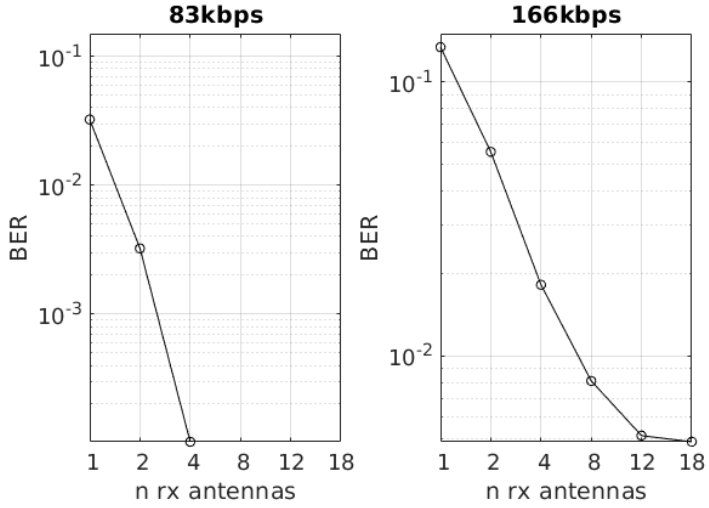

Figure 33: 3x3 DUSTM BER with symbol timing synchronization.

The realized bandwidth efficiency for both of the schemes evaluated here reduces to

$$\eta = \frac{|\mathcal{G}|/p}{1+\beta} \tag{66}$$

For the 2×2 code, $\eta \approx 1.1$ bit/s/Hz and for the 3×3 code, $\eta \approx 1$ bit/s/Hz. These performance metrics are not phenomenal, there is still room for improvement.

The measured signal quality in these experiments varied from 1 dB to 8 dB SNR between receive antennas. The estimated SNR for the 2×2 transmission is presented in Table 3. These values are only informal estimates: they were computed by comparing against a snapshot of data after the matched filter without transmission. In the BER measurements, any kind of switched combining for the best quality signals was not employed, receive antennas were added trivially in numerical order. Finally, the measured power spectral density from 10 antennas for the 125 ksym/s

Table 3: Sample SNR for 2×2 DUSTM, 8 samples per symbol

channel	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
SNR [dB]	6	8	6	6	7	5	2	6	5
channel	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
SNR [dB]	7	4	2	1	3	2	6	3	6

transmission is visualized in Figure 34.

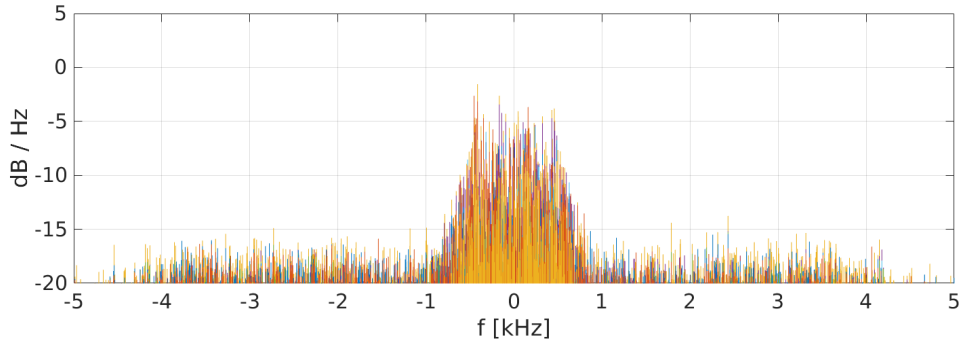

Figure 34: Periodogram of received spectrum for 2x2 DUSTM

6 Summary

This work was conducted utilizing highly non-ideal repurposed radio hardware. In spite of this, an array of inexpensive SDRs was successfully brought to phase coherence. The tests showed that the implementation works, despite the shortcomings of the noise source. The receiver built here was put to use in experiments on direction of arrival estimation and MIMO space-time coding.

A 2×2 differential space-time block code was successfully received at bit rates of 94 kbps and 188 kbps. Extended to three antenna 3×3 code, the bit rates were 83 kbps and 166 kbps. Maximum number of receive antennas was 18 and the lower bit rate yielded 0 bit error rate for both codes after 8 receiver antennas. The experiments were conducted with quite poor received signal quality. Hence, the overall transmission rates realized here are not remarkable by today's standards, but serve to prove that the schemes and the receiver work. The receiver concept is of more importance in and of itself.

We find that multiple RTL-SDR can be time synchronized over the full tuning range of the device, up to 1.7 GHz. However, phase coherence requires correction coefficients due to the uncertainty of the tuner LO phase. These corrections can only be computed within the reach of the reference noise, which in the current prototype is limited below 1.5 GHz. Furthermore, the LO phase was discovered to occasionally drift over the course of an hour, meaning that the correction coefficients may have to be periodically recalculated in continuous operation.

Ideally, the tuner LO signal for all dongles would be generated from a single shared source. Such a setup might still need phase correction due to component differences, but the correction coefficients would need to be calibrated far less often. However, this is unimplementable with the monolithic R820T/2 tuner.

The receiver software could be further optimized by revising the synchronization algorithm to reduce computational load. Current version of the software was tested to be able to handle at least 35 RTL-SDR dongles on a typical desktop PC. Moreover, the source code is still too much a cross between C and C++, since the initial proof-of-concept implementation was written in plain C. The source code is available for download via *git* from `https://github.com/mlaaks/coherentsdr`. The project directory also contains the noise source schematic.

In a future revision of the noise source, the signal switching must be improved to mitigate the leaking at certain frequencies. Perhaps the whole amplifier chain should be switched off when reference is not needed, not just the output stages. Future work and extensions to the implementation could also include bandwidth aggregation to increase the sampling rate. That is, joining the samples from multiple RTL-SDR sharing a single antenna by upsampling and frequency translation. It is not certain how well this would work.

References

[1] M. Da Silva and F. Monteiro, *MIMO Processing for 4G and Beyond.* Bosa Roca: CRC Press, 2016.

[2] B. Vucetic and J. Yuan, *Space-time coding.* Chichester, England: Wiley & Sons, 2003.

[3] S. M. Alamouti, "A simple transmit diversity technique for wireless communications," in *IEEE Journal on Selected Areas in Communications,* vol. 16, no. 8, pp. 1451-1458, Oct. 1998.

[4] B. Hughes, "Differential Space-Time Modulation," *IEEE Transactions on Information Theory,* vol. 46, no.7, NOV. 2000.

[5] Ettus Research, "USRP Software Defined Radio (SDR) online catalog." Internet: https://www.ettus.com/product [Jan. 24, 2019].

[6] S. Malkowsky et al., "The World's First Real-Time Testbed for Massive MIMO: Design, Implementation and Validation," in *IEEE Access*, vol. 5, pp. 9073-9088, 2017.

[7] S. Markgraf, "RTL-SDR wiki." Internet: https://osmocom.org/projects/rtl-sdr/wiki/Rtl-sdr#rtl-sdr, May. 14, 2018 [Sep. 27, 2018].

[8] J. Vierinen, "Passive radar with $16 dual coherent channel rtlsdr dongle receiver." Internet: http://kaira.sgo.fi/2013/09/passive-radar-with-16-dual-coherent.html, Sep. 26, 2013 [Jan. 13, 2019].

[9] T. Peltola, "Synchronized RTL-SDR Receivers." Internet: https://github.com/tejeez/rtl_coherent, Jul. 6, 2016 [Sep. 29, 2018].

[10] R. Cerda, "Pierce-Gate Crystal Oscillator, an introduction", *Microwave Product Digest.* Mar. 2018. Accessed on: Oct. 3, 2018. [Online]. Available: http://www.crystek.com/documents/appnotes/pierce-gateintroduction.pdf

[11] S. Arslan and B. S. Yıldırım, "A Broadband Microwave Noise Generator Using Zener Diodes and a New Technique for Generating White Noise," *IEEE Microwave and Wireless Components Letters,* vol. 28, no. 4, pp. 329-331, Apr. 2018.

[12] P. I. Somlo, "Zener-diode noise generators," *Electronics Letters,* vol. 11, no. 14, pp. 290-, 10 Jul. 1975.

[13] D. E. Susans, "Noise calibrator for v.h.f. and u.h.f. field-strength-measuring receivers," *Electronics Letters,* vol. 3, no. 8, pp. 354-355, Aug. 1967.

[14] S. Markgraf, "osmo-fl2k." Internet: https://osmocom.org/projects/osmo-fl2k/wiki, May. 14, 2018 [Jan. 10, 2018].

[15] R. Schmidt, "Multiple emitter location and signal parameter estimation," *IEEE Transactions on Antennas and Propagation,* vol.34, no.3, pp.276-280, Mar. 1986.

[16] M. Rice, *Digital Communications.* Upper Saddle River: Pearson Prentice Hall, 2009.

[17] E. Krouk and S. Semenov, *Modulation and coding techniques in wireless communications.* Chichester, England: Wiley & Sons, 2011.

[18] M. Jankiraman, *Space-time codes and MIMO systems.* Boston: Artech House, 2004.

[19] J. Salz and J. H. Winters, "Effect of fading correlation on adaptive arrays in digital wireless communications," in *Proceedings of ICC '93 - IEEE International Conference on Communications,* Geneva, Switzerland, 1993, pp. 1768-1774 vol.3.

[20] J. Oppenheim and B. Reznik, "Probabilistic and information-theoretic interpretation of quantum evolutions", *Physical Review,* A. 70, 2004.

[21] N. Wheeler "Relationships among the Unitary Bases of Weyl, Schwinger, Werner & Oppenheim." Internet: https://www.reed.edu/physics/faculty/wheeler/documents/Miscellaneous%20Math/Unitary%20Bases.pdf, Mar. 2012 [Dec. 14, 2018].

[22] R. Adve, "Direction of Arrival Estimation" Internet: https://www.comm.utoronto.ca/~rsadve/Notes/DOA.pdf [Jan. 26, 2018].

[23] D. Linebarger, R. DeGroat, E. Dowling, P. Stoica and G. Fudge, "Incorporating a priori information into MUSIC-algorithms and analysis," *Signal Processing,* vol. 46, no. 1. pp. 85-104, 1995.

[24] R. Stewart, K. Barlee, D. Atkinson and L. Crockett, *Software Defined Radio using MATLAB & Simulink and the RTL-SDR.* Glasgow, Strathclyde Academic Media, 2015. Available: www.desktopSDR.com [Dec 11, 2018].

[25] M. Kuhn, "Compromising emanations: eavesdropping risks of computer displays," University of Cambridge, Rep. no. 577, Dec. 2003. Accessed on: Jan. 2, 2018. [Online]. Available: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-577.pdf

[26] A. Oppenheim, R. Schafer, *Discrete-Time Signal Processing.* Upper Saddle River (N.J.): Prentice-Hall, 2010.

[27] D. Elliot, K. Rao, *Fast Transform Algorithms, Analyses, Applications.* London: Academic Presee, 1983.

[28] W. Press, S. Teukolsky, W. Vetterling and B. Flannery, *Numerical Recipes : The Art of Scientific Computing.* 3rd edition. New York: Cambridge University Press, 2007.

[29] W. Woyczyński, *A First Course in Statistics for Signal Analysis.* Boston, Birkhäuser, 2006.

[30] L. Sliwczynski, "Zener diode and MMICs produce true broadband noise." Internet: https://www.edn.com/design/test-and-measurement/4358938/Zener-diode-and-MMICs-produce-true-broadband-noise, Oct. 14, 1999 [Oct. 3,2018].

[31] Fairchild Semiconductor, "P-Channel 1.8V Specified PowerTrench MOSFET," FDN304P datasheet, Jan. 2001.

[32] P. Hintjens, "ZeroMQ guide." Internet: http://zguide.zeromq.org/page:all, [Dec. 17, 2018].

[33] T. Peltola, "Experimental librtlsdr branch." Internet: https://github.com/tejeez/rtl-sdr, Nov. 1, 2017 [Sep. 29, 2018].

[34] M. Frigo and S. Johnson, "Fastest Fourier Transform in the West." Internet: http://www.fftw.org/#documentation, Oct. 30, 2018 [Sep. 15, 2018].

[35] N. West, "Vector Optimized Library of Kernels." Internet: http://libvolk.org/, [Sep. 10, 2018].

[36] *Universal Serial Bus Specification,* Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC and Philips, Apr. 27, 2000 [Online]. Available: https://usb.org/document-library/usb-20-specification

[37] S. McGill, "Zeromq-matlab" Internet: https://github.com/smcgill3/zeromq-matlab, Feb. 3, 2015 [Jan. 2, 2018].

[38] T. I. Laakso, V. Valimaki, M. Karjalainen and U. K. Laine, "Splitting the unit delay [FIR/all pass filters design]," *IEEE Signal Processing Magazine,* vol. 13, no. 1, pp. 30-60, Jan. 1996.

[39] Texas Instruments, "SNx4HC04 Hex Inverters," 74HC04 datasheet, Dec. 1982 [Revised Sept. 2015].

[40] F. Olver, D. Lozier, R. Boisvert and C. Clark, *NIST Handbook of Mathematical Functions.* Cambridge University Press, 2010.

A Circuit diagram of noise- and clock generators