

# Bachelorarbeit

im  
Studiengang Electrical and Electronic Engineering (ISE)(B.Sc.)

---

## OPTIMIERUNG UND WEITERENTWICKLUNG EINES MIMO-AUDIODEMONSTRATORS

---

angefertigt von  
Gengqi Liu

bei  
Prof. Dr.-Ing. A. Czylwik

Fachgebiet  
Nachrichtentechnische Systeme

an der  
Universität Duisburg-Essen

Duisburg, Februar 2026

## Abstract

This thesis presents the design and implementation of a multi-channel MIMO-OFDM audio demonstrator intended for teaching and experimental analysis of physical-layer communication concepts. The work focuses on adapting an existing software framework to a modern multi-channel audio interface, restructuring the graphical user interface using MATLAB App Designer, and implementing a configurable MIMO-OFDM receiver chain that exposes relevant intermediate processing results.

The demonstrator enables end-to-end transmission of payload data, such as text, over an acoustic multi-input multi-output channel using spatial multiplexing and OFDM modulation. Frame-based audio playback and recording are employed to ensure stable operation, while receiver processing includes timing synchronization, carrier-frequency-offset compensation, channel estimation, MIMO equalization, symbol demapping, and payload reconstruction.

Beyond payload recovery, the system provides access to a range of physical-layer observables, including synchronization metrics, estimated channel impulse and transfer functions, equalized symbol constellations, channel rank measures, and error vector magnitude (EVM). These quantities allow systematic evaluation of system behavior and facilitate intuitive visualization of MIMO-OFDM principles.

Experimental results obtained with a 4-transmit, 8-receive spatial multiplexing configuration using 4-QAM modulation demonstrate stable operation and reproducible analysis outputs. While channel coding is not yet integrated at the receiver side, the implemented framework provides a solid basis for further extensions and serves as an effective platform for education-oriented experimentation and analysis of MIMO-OFDM systems.

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Theoretical Background</b>	<b>5</b>
2.1	Radio Propagation Channels and Motivation for OFDM . . . . .	5
2.2	Principles of OFDM Transmission . . . . .	6
2.2.1	Discrete-Time OFDM Signal Model . . . . .	7
2.2.2	Cyclic Prefix and Circular Convolution . . . . .	8
2.2.3	Frequency-Domain Representation and One-Tap Equalization . . . . .	8
2.2.4	Advantages and Limitations of OFDM . . . . .	8
2.3	Fundamentals of MIMO Systems . . . . .	9
2.3.1	Motivation for Multi-Antenna Communication . . . . .	9
2.3.2	Transmission Modes in MIMO Systems . . . . .	9
2.3.3	MIMO Channel Model and Spatial Degrees of Freedom . . . . .	10
2.4	MIMO-OFDM Signal Model . . . . .	10
2.4.1	Conceptual Combination of MIMO and OFDM . . . . .	10
2.4.2	Discrete-Time Transmit Model . . . . .	11
2.4.3	Per-Subcarrier MIMO Channel Model . . . . .	11
2.4.4	Structure and Interpretation of the Channel Matrix . . . . .	11
2.4.5	Detection Problem in MIMO-OFDM Systems . . . . .	12
2.5	Channel Estimation and Linear Equalization in the Implemented MIMO-OFDM Receiver . . . . .	12
2.5.1	Training-based channel estimation using a deterministic preamble .	12
2.5.2	Linear MMSE equalization on each subcarrier . . . . .	13
2.5.3	Relation to observable receiver metrics . . . . .	13
<b>3</b>	<b>Methodology: System Adaptation and Development</b>	<b>14</b>
3.1	System-level workflow and design principles . . . . .	14
3.2	Objective 1: Adapting the software to the new multi-channel audio interface . . . . .	15
3.2.1	Hardware platform upgrade and its implications . . . . .	15
3.2.2	Software-level audio I/O interface and channel mapping . . . . .	16
3.2.3	Frame-based streaming, underrun/overrun monitoring, and fallback mode . . . . .	17
3.2.4	Amplitude protection and traceability via audio logging . . . . .	17
3.3	Objective 2: Porting and restructuring the GUI using MATLAB App Designer . . . . .	18
3.3.1	Motivation for migration and architectural objectives . . . . .	18
3.3.2	Event-driven workflow and separation of concerns . . . . .	18
3.3.3	Graphical layout as a reflection of the signal-processing pipeline .	19
3.3.4	Global parameter management and state consistency . . . . .	19
3.3.5	GUI-driven configuration of transmission and reception methods .	20
3.3.6	Structured parameter interface via <code>procParam</code> . . . . .	20
3.3.7	Methodological implications . . . . .	21
3.4	Objective 3: End-to-end PHY pipeline for text transmission . . . . .	21
3.4.1	Processing contract and data interface ( <code>procParam</code> → <code>data</code> ) . . . . .	22
3.4.2	Receiver front-end: baseband conversion, timing, and OFDM extraction . . . . .	22

3.4.3	Training structure and per-subframe parsing for spatial multiplexing and V-BLAST . . . . .	24
3.4.4	Training-based channel estimation used in the receiver (ZF/LS via preamble division) . . . . .	24
3.4.5	Noise power and SNR proxy from the trailing blocks . . . . .	24
3.4.6	Per-subcarrier MMSE detection and optional decision-directed phase tracking . . . . .	24
3.4.7	Symbol demapping and ASCII reconstruction . . . . .	25
3.4.8	Stored observables for GUI-based analysis . . . . .	25
3.5	Chapter summary . . . . .	26
<b>4</b>	<b>Results</b>	<b>26</b>
4.1	Recorded transmit and receive signals . . . . .	26
4.2	Timing synchronization results . . . . .	27
4.3	Channel estimation outputs: CIR and CTF . . . . .	28
4.4	Equalized symbol-domain results: constellations and EVM . . . . .	29
4.4.1	Equalized constellations per spatial stream . . . . .	29
4.4.2	EVM-based quality metrics . . . . .	29
4.5	Channel rank and conditioning results . . . . .	30
4.5.1	Rank over subcarriers and rank histogram . . . . .	30
4.5.2	Minimum singular value and condition-number proxy . . . . .	31
4.6	End-to-end payload reconstruction and bit error results . . . . .	31
4.6.1	Recovered text output . . . . .	32
4.6.2	Bit error count and BER (GUI indicators) . . . . .	32
4.7	Key metrics summary (tool-reported snapshot) . . . . .	33
4.8	From results to discussion . . . . .	33
<b>5</b>	<b>Discussion</b>	<b>33</b>
5.1	Reliable decoding despite dispersed constellations . . . . .	34
5.2	Influence of cyclic prefix length on constellation quality . . . . .	34
5.3	Global frame start selection and CIR structure . . . . .	35
5.4	Constellation contraction at low transmit amplitude . . . . .	35
5.5	Engineering implications and refinement options . . . . .	36
<b>6</b>	<b>Conclusion</b>	<b>36</b>
6.1	Summary of achieved objectives . . . . .	37
6.2	Key outcomes and insights . . . . .	38
6.3	Consolidation and future perspectives . . . . .	38

# 1 Introduction

Efficient data transmission over frequency-selective and multipath channels has long challenged digital communication systems. One widely adopted solution to this problem is *Orthogonal Frequency-Division Multiplexing* (OFDM), which mitigates inter-symbol interference (ISI) by distributing a high-rate data stream over a large number of orthogonal narrowband subcarriers and by employing a cyclic prefix [1, 2]. Due to its robustness against multipath propagation, OFDM has become a key modulation technique for broadband communication systems.

In parallel, *Multiple-Input Multiple-Output* (MIMO) technology exploits multiple transmit and receive antennas to increase channel capacity or improve link reliability without requiring additional bandwidth or transmit power. Fundamental theoretical works have shown that MIMO systems can significantly enhance spectral efficiency by exploiting spatial degrees of freedom [3, 4]. By combining MIMO with OFDM, MIMO-OFDM systems are able to jointly exploit spatial, frequency, and temporal diversity, making them particularly suitable for broadband and frequency-selective channels.

As a result of these advantages, MIMO-OFDM has been widely adopted in modern wireless communication standards, including IEEE 802.11n/ac/ax and cellular systems such as LTE and 5G [5]. Consequently, MIMO-OFDM represents not only a topic of high theoretical relevance, but also a core technology in practical communication systems. This importance makes it an essential subject in communication engineering education.

Beyond conventional radio-frequency applications, MIMO-OFDM has also been investigated for use in acoustic transmission environments, most notably in underwater acoustic communication systems. Acoustic channels are characterized by limited available bandwidth, low propagation speed, severe multipath propagation, and pronounced Doppler effects, all of which pose significant challenges for reliable high-rate communication [6]. Several studies have demonstrated that MIMO-OFDM can substantially improve spectral efficiency and robustness in such environments when appropriate synchronization and channel estimation techniques are employed [7, 8].

In the context of communication education, experimental platforms based on radio-frequency hardware are often complex and costly, and they typically offer limited possibilities for intuitive perception of the transmission process. An alternative approach is provided by acoustic MIMO demonstrators, in which loudspeakers and microphones are used instead of conventional radio transmitters and receivers. This approach enables low-cost experimentation while allowing the transmitted signals to be directly audible, thereby enhancing the intuitive understanding of signal processing algorithms.

Within the EIT study program, an existing acoustic MIMO demonstrator has been developed to illustrate fundamental concepts of MIMO signal transmission and reception. The system is implemented in MATLAB and realizes a complete signal chain from signal generation and modulation to acoustic transmission, reception, and baseband processing. The original system was designed for earlier audio hardware and employed MATLAB's GUIDE framework for the graphical user interface. While the demonstrator already enables the presentation of basic MIMO concepts, its structure and functionality exhibit limitations with respect to modern hardware interfaces, software maintainability, and extensibility.

In particular, the replacement of the original audio hardware by a modern multi-channel audio interface requires adaptations in signal generation, acquisition, and channel mapping. Furthermore, the GUIDE-based graphical user interface does not support a

modular and extensible design suitable for future developments. From an algorithmic perspective, the existing demonstrator provides only limited insight into receiver-side signal processing and lacks comprehensive visualization of key performance metrics such as channel characteristics, spatial degrees of freedom, channel rank, signal-to-noise ratio (SNR), and error vector magnitude (EVM).

Against this background, the objective of this bachelor thesis is the systematic optimization and further development of the existing acoustic MIMO demonstrator. This includes adapting the software to the new hardware platform, redesigning the graphical user interface using MATLAB App Designer, and optimizing receiver-side signal processing algorithms. In addition, new analysis and visualization features are introduced to enhance the demonstrator's functionality and to provide a clearer and more comprehensive illustration of MIMO-OFDM principles for educational and experimental purposes.

## 2 Theoretical Background

### 2.1 Radio Propagation Channels and Motivation for OFDM

In practical communication environments, signal propagation between transmitter and receiver is rarely limited to a single line-of-sight path. Instead, transmitted signals are reflected, diffracted, and scattered by surrounding objects, resulting in multiple propagation paths with different delays and attenuations. This phenomenon is commonly referred to as multipath propagation. Under the assumption that the channel remains constant over the observation interval, the propagation channel can be modeled as a linear time-invariant (LTI) system [9].

In this case, the channel is fully characterized by its channel impulse response (CIR), which can be expressed as

$$h_c(t) = \sum_{l=1}^L h_{c,l} \delta(t - \tau_l), \quad (1)$$

where  $h_{c,l}$  and  $\tau_l$  denote the complex-valued path coefficient and propagation delay of the  $l$ -th multipath component, respectively, and  $L$  represents the total number of propagation paths. The corresponding channel transfer function (CTF) in the frequency domain is obtained by the Fourier transform of the CIR and is given by

$$H_c(\omega) = \int_{-\infty}^{\infty} h_c(t) e^{-j\omega t} dt = \sum_{l=1}^L h_{c,l} e^{-j\omega \tau_l}. \quad (2)$$

This representation highlights the frequency-selective nature of multipath channels and is commonly used in the analysis of multicarrier transmission systems [2].

A statistical characterization of the multipath channel is provided by the power delay profile (PDP), defined as the expected squared magnitude of the CIR:

$$P_{\text{PDP}}(\tau) = \mathbb{E}\{|h_c(\tau)|^2\}. \quad (3)$$

The PDP describes how the received signal energy is distributed over different delays and allows the definition of the maximum excess delay  $\tau_{\max}$ . Based on this parameter, the coherence bandwidth  $B_c$  of the channel can be approximated as

$$B_c \approx \frac{1}{\tau_{\max}}. \quad (4)$$

If the signal bandwidth  $B$  is much smaller than  $B_c$ , the channel can be regarded as frequency-flat. Conversely, if  $B$  exceeds  $B_c$ , the channel exhibits frequency-selective fading [9].

In the time domain, frequency-selective channels give rise to inter-symbol interference (ISI), since delayed replicas of previously transmitted symbols overlap with the current symbol. In single-carrier transmission systems, the mitigation of ISI typically requires sophisticated time-domain equalization techniques whose complexity increases significantly with the channel delay spread [2]. This complexity motivates the use of alternative transmission schemes that can more efficiently cope with frequency-selective channels.

The fundamental idea of multicarrier transmission is to decompose a wideband frequency-selective channel into a set of narrowband subchannels, each of which experiences approximately flat fading. Orthogonal Frequency Division Multiplexing (OFDM) represents a practical and efficient realization of this concept. By transmitting data symbols in parallel over a large number of orthogonal subcarriers, OFDM significantly increases the symbol duration on each subcarrier and thereby reduces the impact of ISI [1].

In OFDM systems, the subcarrier spacing  $\Delta f$  is chosen as

$$\Delta f = \frac{1}{T_s}, \quad (5)$$

where  $T_s$  denotes the useful OFDM symbol duration. This choice ensures orthogonality among the subcarriers under ideal synchronization conditions. To preserve subcarrier orthogonality in the presence of multipath propagation, a cyclic prefix (CP) is inserted at the beginning of each OFDM symbol. If the CP length is greater than or equal to the maximum channel delay spread, linear convolution with the channel impulse response is transformed into circular convolution. As a result, the frequency-domain channel matrix becomes diagonal, enabling simple one-tap equalization on each subcarrier [9, 1].

For discrete-time baseband modeling, the multipath channel is commonly represented as a finite impulse response (FIR) filter, also referred to as a tapped delay line model. This discrete-time representation provides the mathematical foundation for the implementation of OFDM systems using inverse discrete Fourier transform (IDFT) and discrete Fourier transform (DFT) operations, and it forms the basis for most practical OFDM transmitters and receivers.

Figure 1 illustrates a generic block diagram of an OFDM transmitter and receiver, highlighting the key signal-processing stages discussed above. In particular, the diagram emphasizes how time and frequency synchronization, cyclic prefix handling, and frequency-domain equalization interact to mitigate the effects of multipath propagation and enable efficient multicarrier transmission.

## 2.2 Principles of OFDM Transmission

Orthogonal Frequency Division Multiplexing (OFDM) is a multicarrier transmission technique whose fundamental idea is to decompose a wideband frequency-selective channel into a set of mutually orthogonal narrowband subchannels. By doing so, each subchannel can be approximated as a frequency-flat channel, which significantly reduces the complexity of receiver-side equalization.

In practical systems, the implementation of OFDM relies on discrete-time signal processing, in particular on the discrete Fourier transform (DFT) and its fast algorithm

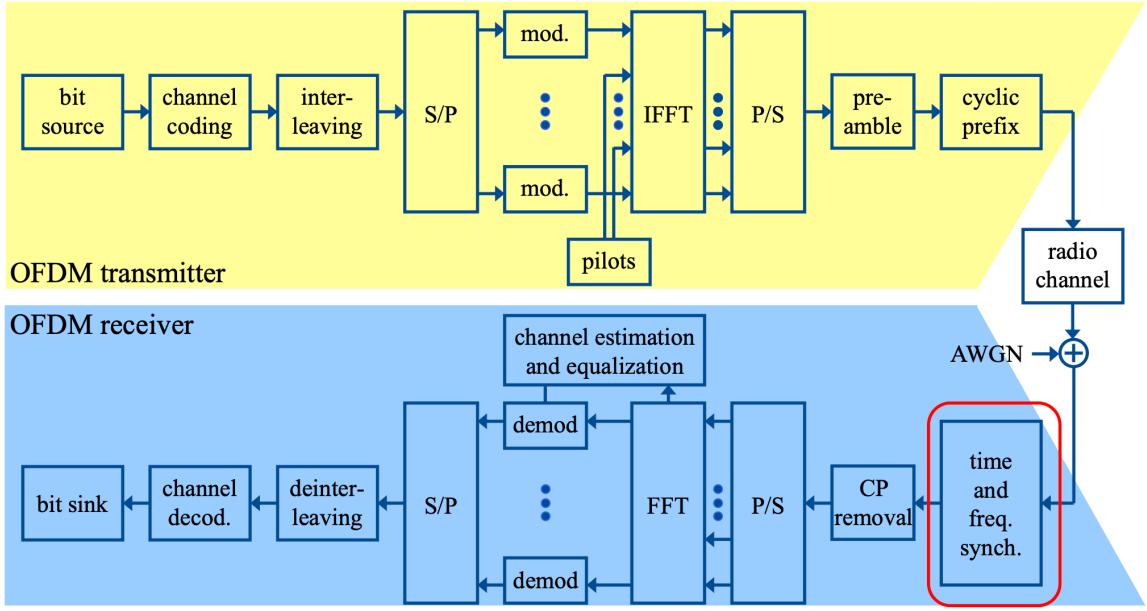


Figure 1: Block diagram of a generic OFDM transmitter and receiver chain, illustrating the main signal-processing stages and the role of time and frequency synchronization. Adapted from [9].

(FFT). Compared to early multicarrier schemes based on analog oscillators, OFDM enables the realization of orthogonal subcarriers through digital signal processing, making it highly suitable for practical communication systems [1, 9].

### 2.2.1 Discrete-Time OFDM Signal Model

Consider a block of complex-valued data symbols

$$\{X[k]\}_{k=0}^{N-1},$$

to be transmitted over  $N$  subcarriers within one OFDM symbol. These frequency-domain symbols are mapped onto orthogonal subcarriers and transformed into the time domain by means of an inverse discrete Fourier transform (IDFT). The resulting discrete-time OFDM signal can be expressed as

$$x[n] = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X[k] e^{j2\pi kn/N}, \quad n = 0, 1, \dots, N-1. \quad (6)$$

The normalization factor  $1/\sqrt{N}$  ensures that the average signal power is preserved during the transformation. In practice, the IDFT operation is efficiently implemented using the inverse fast Fourier transform (IFFT).

The orthogonality of the subcarriers is guaranteed by an appropriate choice of the subcarrier spacing. Let  $T_s$  denote the useful OFDM symbol duration. The subcarrier spacing is then defined as

$$\Delta f = \frac{1}{T_s}. \quad (7)$$

Under ideal synchronization conditions, this choice ensures that the demodulation of one subcarrier does not introduce interference from other subcarriers.

### 2.2.2 Cyclic Prefix and Circular Convolution

When an OFDM signal is transmitted over a multipath channel, the transmitted signal undergoes linear convolution with the channel impulse response. Without further measures, this linear convolution destroys the orthogonality among subcarriers and leads to inter-symbol and inter-carrier interference.

To avoid this effect, a cyclic prefix (CP) is inserted at the beginning of each OFDM symbol. The CP is generated by copying the last  $N_{\text{CP}}$  samples of the OFDM symbol and appending them to its front.

If the CP length satisfies

$$N_{\text{CP}} \geq L_h - 1, \quad (8)$$

where  $L_h$  denotes the length of the discrete-time channel impulse response, the linear convolution can be transformed into circular convolution after CP removal. In this case, the received discrete-time signal can be written as

$$y[n] = x[n] \circledast h[n] + w[n], \quad (9)$$

where  $\circledast$  denotes circular convolution and  $w[n]$  represents additive noise [2].

### 2.2.3 Frequency-Domain Representation and One-Tap Equalization

Applying the discrete Fourier transform to the received OFDM symbol yields a frequency-domain signal model of the form

$$Y[k] = H[k] X[k] + W[k], \quad k = 0, 1, \dots, N - 1. \quad (10)$$

This expression shows that, under ideal conditions, the equivalent channel in the frequency domain exhibits a diagonal structure. Each subcarrier is affected only by the corresponding frequency-domain channel coefficient  $H[k]$ , which allows independent processing of each subcarrier.

A simple equalization method is the zero-forcing (ZF) equalizer, given by

$$\hat{X}[k] = \frac{Y[k]}{H[k]}, \quad (11)$$

provided that  $H[k] \neq 0$ . In the presence of noise, more advanced techniques such as minimum mean square error (MMSE) equalization can be employed to achieve a better trade-off between noise enhancement and distortion [2].

### 2.2.4 Advantages and Limitations of OFDM

The main advantage of OFDM lies in its ability to efficiently combat frequency-selective fading with relatively low receiver complexity. By converting a frequency-selective channel into multiple parallel flat-fading subchannels, OFDM enables simple frequency-domain equalization and supports flexible allocation of spectral resources [1].

Despite these advantages, OFDM also exhibits certain limitations. The superposition of many subcarriers results in a high peak-to-average power ratio (PAPR), which imposes stringent linearity requirements on the transmit power amplifier. Furthermore, OFDM systems are sensitive to synchronization errors, such as carrier frequency offset and timing misalignment. These impairments destroy subcarrier orthogonality and introduce inter-carrier interference (ICI), which must be carefully addressed in practical system implementations and experimental demonstrators [9].

## 2.3 Fundamentals of MIMO Systems

### 2.3.1 Motivation for Multi-Antenna Communication

In conventional single-input single-output (SISO) communication systems, performance is fundamentally limited by channel fading and available bandwidth. In multipath propagation environments, random fluctuations in signal amplitude and phase may lead to deep fades, resulting in a significant degradation of link reliability.

Multi-antenna communication techniques address these limitations by introducing multiple antennas at the transmitter and/or receiver, thereby creating additional spatial degrees of freedom. Multiple-input multiple-output (MIMO) systems exploit these spatial dimensions to either improve link reliability through diversity or increase data rates through parallel transmission, without requiring additional bandwidth or transmit power. For this reason, MIMO has become a key enabling technology in modern broadband communication systems.

From a system-level perspective, the use of multiple antennas naturally generalizes single-antenna configurations. Depending on the number of transmit and receive antennas, systems may operate in SISO, SIMO, MISO, or full MIMO configurations. In the context of the audio-based demonstrator developed in this work, all of these configurations can be realized, with the focus placed on full MIMO operation.

### 2.3.2 Transmission Modes in MIMO Systems

While the presence of multiple antennas provides additional spatial degrees of freedom, the manner in which these degrees of freedom are exploited depends on the chosen transmission mode. Different MIMO transmission strategies emphasize different performance objectives, such as reliability, throughput, or robustness against channel impairments.

**Spatial Multiplexing (Primary Focus of This Work)** Spatial multiplexing aims at increasing the achievable data rate by transmitting multiple independent data streams simultaneously over the same time and frequency resources. Under favorable channel conditions, the achievable throughput can scale approximately linearly with the number of transmit antennas, making spatial multiplexing particularly attractive for spectrally efficient communication.

In spatial multiplexing systems, independent data streams are transmitted from different antennas and superimposed by the propagation channel. At the receiver, multi-antenna signal processing algorithms are required to separate the streams. Common detection techniques include zero-forcing (ZF), minimum mean square error (MMSE) detection, and successive interference cancellation schemes such as the V-BLAST architecture.

For the audio-based MIMO demonstrator considered in this thesis, spatial multiplexing is especially well suited for algorithmic analysis and visualization. It enables a direct illustration of channel rank, spatial degrees of freedom, and receiver detection performance, and is therefore selected as the primary transmission mode for implementation and evaluation.

**Diversity-Oriented Transmission and Alamouti Coding** In contrast to spatial multiplexing, diversity-oriented transmission schemes aim at improving link robustness rather than increasing data rate. A well-known example is the Alamouti space-time block

code, which achieves full transmit diversity for two transmit antennas while retaining a simple linear receiver structure.

In this work, Alamouti transmission is considered mainly as a reference scheme. It provides a useful contrast to spatial multiplexing by highlighting the fundamental trade-off between reliability and throughput in different MIMO transmission modes.

**Eigenmode Transmission and Beamforming** If accurate channel state information is available at the transmitter, the MIMO channel matrix can be decomposed into orthogonal spatial subchannels using singular value decomposition. This principle, commonly referred to as eigenmode transmission or beamforming, enables capacity-optimal transmission under idealized conditions.

Due to its reliance on transmitter-side channel knowledge and increased implementation complexity, eigenmode transmission is not pursued as a primary operating mode in the demonstrator. Nevertheless, it is briefly introduced here to complete the conceptual overview of MIMO transmission strategies.

### 2.3.3 MIMO Channel Model and Spatial Degrees of Freedom

For analytical purposes, MIMO systems are commonly described using a matrix-based baseband-equivalent channel model. Assuming  $N_T$  transmit antennas and  $N_R$  receive antennas, the input–output relationship can be written as

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n}, \quad (12)$$

where  $\mathbf{x} \in \mathbb{C}^{N_T \times 1}$  denotes the transmit signal vector,  $\mathbf{y} \in \mathbb{C}^{N_R \times 1}$  the receive signal vector,  $\mathbf{H} \in \mathbb{C}^{N_R \times N_T}$  the channel matrix, and  $\mathbf{n}$  additive noise.

An essential property of the MIMO channel is its rank, which determines the number of independent spatial data streams that can be transmitted simultaneously. The channel rank is bounded by

$$\text{rank}(\mathbf{H}) \leq \min(N_T, N_R). \quad (13)$$

In spatial multiplexing systems, the channel rank directly limits the achievable multiplexing gain. Consequently, rank-related metrics play a central role in the analysis and evaluation of MIMO-OFDM systems and form a key part of the experimental results presented in later chapters.

## 2.4 MIMO-OFDM Signal Model

### 2.4.1 Conceptual Combination of MIMO and OFDM

OFDM enables the decomposition of a frequency-selective wideband channel into a set of parallel frequency-flat subchannels, while MIMO systems exploit spatial degrees of freedom by employing multiple transmit and receive antennas. The combination of these two techniques results in a MIMO-OFDM system, which has become the dominant transmission scheme in modern broadband communication systems [1, 10].

The key idea of MIMO-OFDM is to apply a narrowband MIMO channel model independently to each OFDM subcarrier. By transforming a frequency-selective channel into multiple approximately flat-fading subchannels, OFDM allows the wideband MIMO channel to be represented as a collection of parallel narrowband MIMO channels in the frequency domain. This per-subcarrier representation significantly simplifies both analytical treatment and receiver design [2, 9].

### 2.4.2 Discrete-Time Transmit Model

Consider a MIMO-OFDM system with  $N_T$  transmit antennas and  $N_R$  receive antennas employing  $N$  subcarriers per OFDM symbol. In spatial multiplexing mode,  $N_S$  independent data streams are transmitted simultaneously, where

$$N_S \leq \min(N_T, N_R).$$

On the  $k$ -th subcarrier, the transmitted signal is represented by the vector

$$\mathbf{x}[k] \in \mathbb{C}^{N_T \times 1},$$

whose elements correspond to the complex-valued symbols transmitted from the individual antennas. After OFDM modulation, including IFFT processing and cyclic prefix insertion, the time-domain signals of all transmit antennas are emitted simultaneously and propagated through the spatial channel.

### 2.4.3 Per-Subcarrier MIMO Channel Model

At the receiver, following cyclic prefix removal and FFT processing, the received signal on the  $k$ -th subcarrier can be expressed as

$$\mathbf{y}[k] \in \mathbb{C}^{N_R \times 1}.$$

Under ideal synchronization conditions, the input–output relationship on each subcarrier is described by the linear model

$$\mathbf{y}[k] = \mathbf{H}[k]\mathbf{x}[k] + \mathbf{n}[k], \quad (14)$$

where  $\mathbf{H}[k] \in \mathbb{C}^{N_R \times N_T}$  denotes the frequency-domain MIMO channel matrix and  $\mathbf{n}[k]$  additive noise.

This formulation shows that OFDM converts the convolutional wideband MIMO channel into a set of independent linear MIMO systems in the frequency domain, each of which can be processed separately [2, 10].

### 2.4.4 Structure and Interpretation of the Channel Matrix

Each element  $H_{ij}[k]$  of the channel matrix represents the complex frequency response between the  $j$ -th transmit antenna and the  $i$ -th receive antenna on the  $k$ -th subcarrier. The matrix  $\mathbf{H}[k]$  therefore captures both the multipath propagation effects and the spatial characteristics of the transmission environment.

In spatial multiplexing systems, the rank of  $\mathbf{H}[k]$  determines the number of independent data streams that can be reliably transmitted on the corresponding subcarrier. Full-rank channel realizations allow the system to fully exploit spatial degrees of freedom, whereas rank deficiencies directly limit multiplexing capability. As a result, subcarrier-dependent variations of the channel matrix play a central role in the performance behavior of MIMO-OFDM systems [10].

### 2.4.5 Detection Problem in MIMO-OFDM Systems

The task of the receiver in spatial multiplexing mode is to recover the transmit vector  $\mathbf{x}[k]$  from the received signal  $\mathbf{y}[k]$ . Due to the coupling of multiple data streams through the channel matrix, this constitutes a multidimensional detection problem.

Linear detection schemes such as zero-forcing (ZF) and minimum mean square error (MMSE) detection offer low computational complexity, while successive interference cancellation approaches, including the V-BLAST architecture, improve detection performance at the cost of increased complexity. In MIMO-OFDM systems, detection is typically performed independently on each subcarrier, preserving the modular structure introduced by OFDM.

## 2.5 Channel Estimation and Linear Equalization in the Implemented MIMO-OFDM Receiver

Based on the per-subcarrier MIMO-OFDM signal model introduced in the previous section,

$$\mathbf{y}[k] = \mathbf{H}[k]\mathbf{x}[k] + \mathbf{n}[k], \quad (15)$$

reliable recovery of the transmitted symbol vector  $\mathbf{x}[k]$  requires knowledge of the frequency-domain MIMO channel matrix  $\mathbf{H}[k]$  and an appropriate receiver-side equalization strategy. In the implemented audio-based demonstrator, channel estimation and equalization are therefore central components of the receiver chain, directly determining the quality of all subsequent symbol-domain and bit-level results reported in Chapter 4.

Rather than implementing a broad set of estimation and detection techniques, the demonstrator deliberately focuses on a compact and transparent processing chain that is well suited for analysis and teaching. Consequently, the receiver employs (i) a *training-based least-squares / zero-forcing (LS/ZF)* channel estimator derived from a deterministic preamble, and (ii) a *linear MMSE* equalizer applied independently on each OFDM subcarrier. Both choices are consistent with standard MIMO-OFDM theory and allow a clear connection between mathematical models, implementation, and observable performance metrics.

### 2.5.1 Training-based channel estimation using a deterministic preamble

Channel estimation in the demonstrator is based on a known preamble transmitted prior to the payload data. A deterministic sequence with constant amplitude and favorable correlation properties is used, enabling both robust synchronization and reliable channel estimation. After FFT processing, the preamble provides a known frequency-domain reference on each subcarrier.

Following time synchronization and cyclic-prefix removal, the received training symbols on subcarrier  $k$  can be expressed as

$$\mathbf{Y}[k] = \mathbf{H}[k]\mathbf{X}_p[k] + \mathbf{N}[k], \quad (16)$$

where  $\mathbf{X}_p[k]$  denotes the known training symbol(s) transmitted during the preamble phase. In the implemented framing structure, transmit channels are sounded sequentially, such that on a given training OFDM symbol only one transmit channel is active. For the link between transmit channel  $t$  and receive channel  $r$ , this reduces to

$$Y_{r,t}[k] = H_{r,t}[k] X_p[k] + N_{r,t}[k]. \quad (17)$$

A least-squares estimate of the channel coefficient is obtained by direct inversion of the known training symbol,

$$\hat{H}_{r,t}[k] = \frac{Y_{r,t}[k]}{X_p[k]}. \quad (18)$$

This estimator corresponds to a zero-forcing inversion of the training symbol on each subcarrier and is therefore referred to as “ZF” in the demonstrator configuration. While more sophisticated estimators exist, the LS/ZF approach offers a particularly transparent mapping between the received signal and the estimated channel response, making it well suited for demonstrator-based analysis [2, 10].

The resulting frequency-domain channel estimates are stored as channel transfer functions (CTF). A corresponding channel impulse response (CIR) is obtained by applying an inverse FFT across subcarriers for each loudspeaker–microphone pair, providing an intuitive time-domain view of multipath propagation effects.

### 2.5.2 Linear MMSE equalization on each subcarrier

Once an estimate  $\hat{\mathbf{H}}[k]$  of the channel matrix is available, the receiver proceeds with spatial separation of the simultaneously transmitted data streams. In spatial multiplexing mode, this constitutes a linear detection problem on each subcarrier.

The demonstrator employs a linear minimum mean square error (MMSE) equalizer, which computes the estimate

$$\hat{\mathbf{x}}[k] = \mathbf{W}_{\text{MMSE}}[k]\mathbf{y}[k], \quad (19)$$

with the equalization matrix

$$\mathbf{W}_{\text{MMSE}}[k] = \left( \hat{\mathbf{H}}^H[k] \hat{\mathbf{H}}[k] + \alpha \mathbf{I} \right)^{-1} \hat{\mathbf{H}}^H[k], \quad (20)$$

where  $\alpha > 0$  is a regularization parameter related to the inverse signal-to-noise ratio. Compared to zero-forcing detection, MMSE equalization mitigates excessive noise enhancement on subcarriers where the channel matrix is ill-conditioned, a situation that frequently arises in practical acoustic MIMO channels due to spatial correlation and frequency-selective fading [10].

The MMSE equalizer is applied independently on each subcarrier and OFDM block, producing a set of equalized complex symbols for all spatial streams. These symbols form the basis for constellation visualization, error vector magnitude (EVM) computation, and subsequent bit demapping as reported in Chapter 4.

### 2.5.3 Relation to observable receiver metrics

The combination of LS/ZF channel estimation and linear MMSE equalization establishes a direct link between the theoretical MIMO-OFDM model and the measurable quantities exposed by the demonstrator. Channel impulse and transfer responses reflect the estimated  $\hat{\mathbf{H}}[k]$ , channel rank and conditioning metrics are derived from its singular values, and constellation diagrams as well as EVM results directly visualize the quality of the MMSE-separated symbol streams. In this way, the chosen receiver structure enables a coherent and interpretable mapping from propagation effects to symbol-domain performance.

### 3 Methodology: System Adaptation and Development

The work presented in this chapter translates the signal models and system concepts introduced in Chapter 2 into a concrete, executable acoustic MIMO–OFDM demonstrator. Rather than pursuing implementation optimality in the sense of a real-time communication prototype, the methodology emphasizes reproducibility, transparency, and controlled experimentation.

Three tightly coupled aspects are addressed throughout the chapter. First, the existing software framework is adapted to a modern multi-channel audio interface, enabling synchronized playback and recording across multiple loudspeakers and microphones. Second, the graphical user interface is restructured using MATLAB App Designer to provide a stable orchestration layer for parameter control, execution flow, and result visualization. Third, the physical-layer processing chain is organized such that different MIMO transmission modes and receiver algorithms can be executed, analyzed, and compared under identical conditions.

The focus is therefore not on isolated algorithmic blocks, but on how hardware interfacing, parameter management, and signal processing are combined into a coherent and extensible experimental system.

#### 3.1 System-level workflow and design principles

At the system level, the acoustic MIMO demonstrator follows a closed-loop transmission–reception workflow that mirrors the structure of a conventional digital communication link while remaining explicitly tailored to offline analysis and teaching-oriented use. Each experiment begins with a user-defined configuration of physical-layer (PHY) parameters via the graphical user interface. These parameters determine the waveform structure and receiver processing, including the OFDM frame layout, modulation order, number of transmit and receive channels, selected MIMO mode, and the channel estimation and equalization strategies.

Based on this configuration, a multi-channel transmit waveform is generated in MATLAB, emitted through the audio interface and loudspeaker array, recorded simultaneously by multiple microphones, and subsequently processed in a single receiver pass. The receiver recovers the transmitted payload and, at the same time, exposes a rich set of intermediate observables such as synchronization metrics, channel responses, equalized symbols, and performance indicators. This tight integration between transmission, reception, and analysis is a defining characteristic of the demonstrator.

Within this work, the term *physical-layer (PHY) parameters* refers to all configuration variables that directly influence waveform generation and sample- or symbol-level signal processing. This includes, among others, the FFT size and cyclic prefix length of the OFDM system, the modulation order, the number of transmit and receive channels, the selected MIMO transmission scheme, and the receiver-side estimation and equalization methods. Together, these parameters instantiate the mathematical signal models introduced in Chapter 2 and correspond to the PHY abstraction commonly used in communication system design [2, 10].

A deliberate design decision is the use of *offline* or *nearline* receiver processing. While audio playback and recording are performed in a streaming fashion, all receiver-side signal processing steps—such as synchronization, channel estimation, equalization, and demodulation—are executed only after an entire frame has been captured. As a result, no strict

real-time constraints are imposed on algorithm execution. This allows the use of computationally intensive operations, including FFT-based OFDM demodulation, singular value decomposition in EigenMode operation, and successive interference cancellation in V-BLAST detection, without risking system instability. At the same time, this approach ensures repeatable behavior and facilitates controlled comparisons between different algorithmic configurations.

The audio input/output loop itself must nevertheless operate reliably during streaming. Playback and recording are implemented using frame-based processing with a configurable frame length. In this context, underruns and overruns may occur if the host system cannot supply or retrieve audio data at the required rate, a well-known phenomenon in non-real-time audio systems running on general-purpose operating systems [11, 12]. In the demonstrator, such events are explicitly monitored and reported to the user, but they do not abort the experiment. This behavior reflects the educational orientation of the system: continuity of operation and transparency about system limitations are prioritized over strict real-time guarantees.

A central architectural principle underlying the entire implementation is the strict separation between GUI orchestration and signal-processing functionality. The graphical user interface manages user interaction, parameter validation, execution control, and visualization of results. In contrast, waveform generation, audio input/output, and receiver processing are implemented as standalone MATLAB functions operating on explicitly defined input and output structures. Communication between the GUI and the processing backend is realized exclusively through structured parameter passing rather than implicit shared state.

This modular organization improves code readability and maintainability, simplifies debugging, and allows individual components—such as channel estimators, equalizers, or visualization routines—to be modified or extended without affecting the overall system structure. As a result, the demonstrator can evolve alongside future hardware upgrades and algorithmic extensions while retaining a clear and robust software architecture.

## 3.2 Objective 1: Adapting the software to the new multi-channel audio interface

A prerequisite for meaningful acoustic MIMO experiments is a hardware and software I/O chain that provides *synchronous*, *repeatable*, and *traceable* multi-channel playback and recording. The legacy demonstrator setup relied on loosely integrated measurement and audio components, which made channel-to-channel alignment and reproducibility harder to guarantee and also introduced a number of implicit, partially hard-coded software assumptions. The upgraded platform replaces this setup with a single integrated multi-channel audio interface and a consistent loudspeaker/microphone front-end. As a consequence, the software can treat the audio device as a well-defined multi-channel “RF front-end” with a single sampling clock and explicit channel mapping.

### 3.2.1 Hardware platform upgrade and its implications

The upgraded demonstrator is built around a Focusrite Scarlett 18i20 audio interface, providing multiple synchronized analog input and output channels driven by a single device clock. From a MIMO signal-processing perspective, this is essential: it eliminates inter-device sampling-clock drift and ensures that all transmit and receive channels are aligned

Table 1: Comparison between the legacy and the upgraded hardware platforms used in the acoustic MIMO demonstrator.

Aspect	Legacy setup	Upgraded setup
System integration	NI BNC-2110 with external power supply	Integrated multi-channel audio interface
Clock synchronization	Implicit, wiring-dependent	Hardware-level common device clock
ADC/DAC quality	Basic measurement grade	High-quality audio-grade converters
Loudspeakers	Generic consumer-grade	Genelec 8010 near-field monitors
Microphones	Generic microphones	t.bone SC140 condenser microphones
Linearity and distortion	Limited, noticeable nonlinear effects	Improved linearity and reduced distortion
Software assumptions	Fixed and partially hard-coded	Fully parameterized and configurable

on the same sample grid. Compared to the previous solution based on external supplies and separate routing via a National Instruments BNC-2110, the integrated interface reduces wiring complexity and, more importantly, removes a class of avoidable timing and channel-mismatch issues that would otherwise contaminate MIMO-OFDM observations.

The loudspeaker and microphone front-end was upgraded accordingly. Genelec 8010 active loudspeakers provide a largely linear response and low distortion in the relevant audio band, which improves the interpretability of receiver metrics (e.g., CIR/CTF, EVM) by reducing hardware-induced nonlinear artifacts. The t.bone SC140 condenser microphones offer a comparatively low noise floor and consistent sensitivity across channels, which is particularly beneficial for multi-channel channel estimation and for spatial processing where relative amplitude/phase relationships matter.

Overall, the upgraded platform improves linearity, channel consistency, and synchronization accuracy. This shifts observed impairments toward the actual acoustic propagation environment and the implemented receiver algorithms, rather than toward avoidable hardware and integration artifacts.

### 3.2.2 Software-level audio I/O interface and channel mapping

On the software side, the upgraded hardware is interfaced through MATLAB’s `audioPlayerRecorder` System object, which supports synchronous multi-channel playback and recording on a single device [11]. In the demonstrator, this audio I/O layer is intentionally treated as a narrow and explicit contract between the signal-processing code and the hardware. The contract is defined by the sample rate  $f_s$ , the number of active transmit and receive channels ( $N_T, N_R$ ), the audio frame length  $L$  used for streaming, and the device identifier `deviceName`. In the App Designer implementation, these values are maintained as GUI-controlled app properties (e.g., `app.fDACFreq`, `app.iNoTxAnt`, `app.iNoRxAnt`) and passed explicitly into the I/O function.

Channel mapping is made explicit rather than implicit. In `runScarlettMimoIO`, playback and recording are configured with `PlayerChannelMapping=1:Nt` and `Recorder ChannelMapping=1:Nr`. This design decision directly supports reproducible experiments:

Table 2: Software-level audio I/O parameters defining the interface between MATLAB and the audio hardware.

Parameter	Meaning
$f_s$ ( <code>fs</code> )	Audio sampling rate (common device clock)
$N_T$ ( <code>Nt</code> )	Number of transmit channels (loudspeakers)
$N_R$ ( <code>Nr</code> )	Number of receive channels (microphones)
$L$ ( <code>frameLen</code> )	Frame length (samples per I/O call)
<code>deviceName</code>	Driver/device identifier (platform-dependent)

once the physical cabling is fixed, the software mapping defines a stable interpretation of “Tx1..Tx $N_T$ ” and “Rx1..Rx $N_R$ ” across operating systems and driver configurations. In the GUI workflow, these mappings are consistent with how the transmit waveform is generated (matrix `txSig` with  $N_T$  columns) and how the received waveform is stored (matrix `rx\_raw` with  $N_R$  columns and `app.RxSequence`).

### 3.2.3 Frame-based streaming, underrun/overrun monitoring, and fallback mode

Audio playback and recording are executed using frame-based streaming to the device. The transmit waveform  $\mathbf{x}_{tx} \in \mathbb{R}^{N_{samples} \times N_T}$  is segmented into contiguous blocks of length  $L$  and passed to `audioPlayerRecorder` in a loop. Each I/O call returns the recorded frame  $\mathbf{y}_{rx} \in \mathbb{R}^{L \times N_R}$ , together with underrun/overrun indicators. These events occur if the host system cannot deliver or acquire audio data at the required rate, a known effect in non-real-time audio processing [11, 12]. In the demonstrator, underruns and overruns are surfaced via warnings but do not abort the experiment, reflecting the teaching-oriented goal of maintaining continuity while still making I/O quality visible.

To support development without guaranteed access to the physical device, `runScarlet` `tMimoIO` includes a fallback path: if the device cannot be opened, the system switches to an ideal loopback in which the receive buffer is filled by copying the transmitted channels (up to `min(Nt, Nr)`). This “simulation mode” keeps the end-to-end software pipeline executable and enables debugging of framing, synchronization, and receiver visualization without hardware dependencies. In the App Designer workflow, the device-selection logic is encapsulated (e.g., `app.getAudioDeviceName`), and the transmission callback reports whether the real device or fallback mode was used.

### 3.2.4 Amplitude protection and traceability via audio logging

Because multi-channel audio interfaces and loudspeakers can saturate easily, transmit-side amplitude protection is applied immediately before playback. In `runScarlett` `imoIO`, the waveform is constrained to its real part (the physical audio output is real-valued), peak-normalized to avoid clipping, and scaled by a fixed linear gain (`txGain=0.7`) to provide headroom. This step prevents inadvertent saturation that would otherwise introduce nonlinear distortion and invalidate the interpretation of channel estimates and constellation-based metrics.

For traceability and reproducibility, both transmitted and recorded signals are logged to disk as multi-channel WAV files (`tx\_last.wav`, `rx\_last.wav`). These recordings serve two roles: they enable offline inspection of the raw audio I/O path independent of the receiver chain, and they allow repeated processing of identical recorded data when

validating algorithmic changes. This logging mechanism is therefore treated as part of the experimental methodology rather than a purely auxiliary debugging feature.

### 3.3 Objective 2: Porting and restructuring the GUI using MATLAB App Designer

#### 3.3.1 Motivation for migration and architectural objectives

The original version of the acoustic MIMO demonstrator relied on MATLAB GUIDE for graphical user interface development. As GUIDE has been deprecated and removed from recent MATLAB releases, a migration to MATLAB App Designer was required to ensure long-term maintainability and compatibility with current MATLAB versions [13, 14].

Rather than reproducing the legacy interface on a component-by-component basis, the GUI was fundamentally restructured. The redesign treats the GUI as an orchestration layer for the physical-layer processing chain, instead of a passive visualization front end. In this role, the GUI is responsible for parameter management, execution control, and controlled access to intermediate and final processing results.

This architectural shift reflects the primary purpose of the demonstrator: enabling systematic experimentation and algorithmic analysis. Consequently, the GUI design prioritizes transparency, reproducibility, and extensibility over feature density or real-time interactivity.

Figure 2 provides a global view of the redesigned GUI. Rather than serving as a passive control panel, the interface mirrors the logical structure of the experimental workflow. System configuration, transmission control, receiver inspection, and analytical evaluation are spatially separated, which directly supports reproducible experimentation and structured exploration of PHY-layer behavior.

#### 3.3.2 Event-driven workflow and separation of concerns

The application is implemented as a class-based MATLAB App Designer app, following an event-driven programming model. User interactions are handled through callback functions associated with specific GUI elements, such as buttons, drop-down menus, and numeric edit fields.

A central design principle is the strict separation between control logic and numerical signal processing. GUI callbacks do not perform physical-layer computations themselves. Instead, they validate preconditions, collect user-selected parameters, invoke dedicated processing functions, and store the resulting data structures.

This workflow is exemplified by the callback `StartTransmissionButtonPushed`. When triggered, the callback assembles all relevant system and PHY parameters from the app state, initiates waveform generation and audio I/O, and subsequently triggers receiver-side processing. The complete physical-layer processing chain is executed in external functions, keeping the GUI layer free of algorithm-specific code.

All receiver outputs are stored in the centralized app property `app.RxAnalysisData`. This structure serves as the single source of truth for subsequent analysis and visualization steps, including constellation diagrams, EVM plots, BER evaluation, channel impulse and transfer responses, and channel-rank statistics. By enforcing centralized data storage, the design avoids hidden dependencies between callbacks and ensures consistent access to processing results.

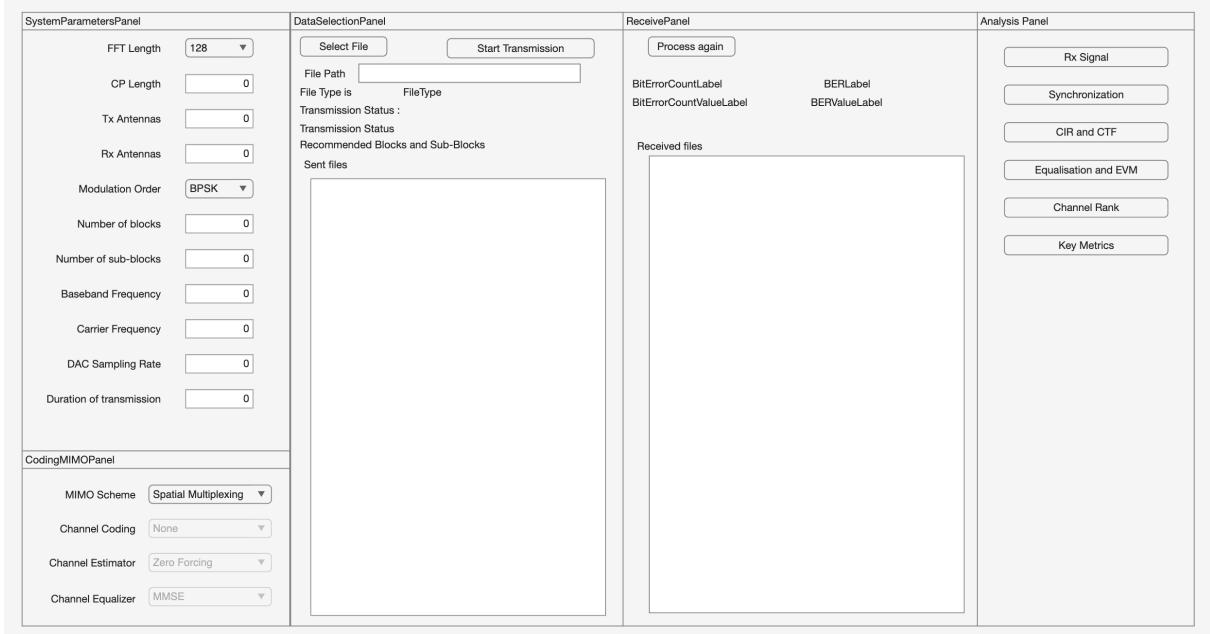


Figure 2: Overview of the MATLAB App Designer–based graphical user interface of the acoustic MIMO demonstrator. The interface is structured into four functional panels: system parameter configuration, data selection and transmission control, receiver output inspection, and analysis and visualization of intermediate PHY metrics. This layout reflects the separation between configuration, execution, and evaluation stages in the experimental workflow.

### 3.3.3 Graphical layout as a reflection of the signal-processing pipeline

The spatial organization of the graphical user interface is intentionally aligned with the structure of the underlying signal-processing pipeline. Rather than grouping controls by implementation detail, the interface is arranged according to the logical stages of a MIMO–OFDM experiment.

System-level and physical-layer parameters are concentrated in the left panel, reflecting the fact that these parameters define the signal model and must be fixed prior to transmission. The central panels are dedicated to data selection, transmission execution, and receiver output inspection, corresponding to the sequential flow from waveform generation to payload recovery. Finally, the analysis panel on the right exposes intermediate PHY metrics such as synchronization results, channel responses, constellation diagrams, EVM, and channel rank.

This left-to-right layout mirrors the causal progression of the processing chain and allows users to relate algorithmic choices directly to observable effects. As a result, the GUI does not merely provide access to functionality, but acts as a visual representation of the end-to-end MIMO–OFDM receiver structure, reinforcing the educational and exploratory objectives of the demonstrator.

### 3.3.4 Global parameter management and state consistency

All system and PHY parameters are maintained as properties of the app object. These include OFDM parameters (FFT length, cyclic prefix length, number of blocks and sub-

Table 3: Representative GUI features supporting methodical experimentation.

Feature group	Implemented functionality
System parameters	FFT size, CP length, $N_T/N_R$ , carrier and sampling rate
Transmission modes	Spatial multiplexing, Alamouti, EigenMode, V-BLAST
Receiver options	Channel estimation and equalization selection
Execution control	Separate triggers for transmission and receiver processing
Analysis tools	Channel responses, constellations, EVM, BER, rank metrics

blocks), MIMO configuration (number of transmit and receive channels, transmission mode), modulation and coding settings, and frequency-related parameters.

Each GUI control element is connected to a dedicated `ValueChangedFcn` callback that performs basic input validation and immediately updates the corresponding internal property. As a result, the internal application state remains consistent at all times, and parameter changes are propagated deterministically throughout the system.

Certain parameter changes additionally trigger dependent updates. For example, modifications to the modulation order or antenna configuration automatically recompute the recommended number of OFDM blocks. Changes to the antenna configuration also invalidate previously estimated EigenMode channel information, which is explicitly cleared from the app state to enforce a new channel sounding step. These mechanisms prevent inconsistent parameter combinations and stale channel-dependent data.

### 3.3.5 GUI-driven configuration of transmission and reception methods

The GUI provides explicit controls for configuring both transmission and reception strategies. Supported MIMO transmission modes include spatial multiplexing, Alamouti coding, V-BLAST, and EigenMode transmission. Channel estimation and equalization methods are selected via drop-down menus and stored as string identifiers within the app state.

The GUI does not interpret these selections internally. Instead, the chosen options are forwarded unchanged to the processing functions via structured parameter passing. This approach keeps the receiver implementation independent of the GUI while still enabling flexible experimentation with different algorithmic configurations.

### 3.3.6 Structured parameter interface via `procParam`

The structured variable `procParam` defines a stable and explicit interface between the GUI layer and the signal-processing backend. It encapsulates all information required for receiver processing, including the recorded waveform, system parameters, algorithm selections, and payload metadata.

By relying on structured parameter passing rather than global variables, the processing functions can be tested and debugged independently of the GUI. This design also simplifies future extensions, as additional parameters can be added to `procParam` without modifying the overall application architecture.

For EigenMode operation, channel-dependent matrices obtained from the sounding phase—such as precoding and decoding matrices and singular values—are included in

Table 4: Key fields of `procParam` used for receiver processing.

Field	Description
<code>rxSignal</code>	Recorded signal matrix [ $N_{\text{samp}} \times N_R$ ]
<code>fs</code>	Sampling rate ( $f_s$ )
<code>iNfft, iNg, iNb</code>	OFDM parameters (FFT size, CP length, block length)
<code>iNoTxAnt, iNoRxAnt</code>	Number of transmit and receive channels
<code>iModOrd</code>	Modulation order (bits per symbol)
<code>mimoMode</code>	Selected MIMO transmission scheme
<code>channelEstimator</code>	Channel estimation method
<code>equalizerMode</code>	Equalization/detection method
<code>DatenTyp, SendeDatei</code>	Payload type and original data

---

**Algorithm 1** GUI-driven execution flow for receiver processing

---

**Require:** Recorded multi-channel signal  $\mathbf{y}_{\text{rx}}$  (if available), GUI configuration state

**Ensure:** Decoded payload and analysis structure `app.RxAnalysisData`

- 1: Check whether a received signal is present; otherwise abort with a user-facing message
  - 2: Assemble the processing parameter structure `procParam` from GUI state  
(OFDM parameters, antenna configuration, modulation, MIMO mode, estimator, equalizer, payload metadata)
  - 3: **if** EigenMode mode is selected **then**
  - 4:     Ensure that the precoder/decoder matrices from channel sounding are available
  - 5:     Attach  $\mathbf{V}_1$  (and optionally  $\mathbf{U}_1, \mathbf{S}_1$ ) to `procParam`
  - 6: **end if**
  - 7: Call the backend processing function `Signalverarbeitung_app(procParam)`
  - 8: Store returned results in `app.RxAnalysisData`
  - 9: Update GUI elements (text preview, BER, and analysis buttons) based on the stored results
- 

`procParam`. This ensures that the receiver processing stage has access to all required channel information in a controlled and reproducible manner.

### 3.3.7 Methodological implications

The event-driven GUI architecture enforces a clear separation between user interaction and physical-layer signal processing. This separation supports systematic experimentation under well-defined conditions and enables individual components of the processing chain to be extended or replaced without modifying the GUI logic itself.

As a result, the GUI functions not merely as a user interface, but as an integral methodological component of the demonstrator. It provides a stable framework in which different algorithms, parameter sets, and transmission modes can be explored in a controlled and reproducible manner.

## 3.4 Objective 3: End-to-end PHY pipeline for text transmission

Objective 3 consolidates the receiver-side PHY chain into a reproducible, inspectable workflow that turns a recorded multi-channel audio frame into a decoded ASCII text payload and a structured set of intermediate observables. The implementation is intentionally

---

**Algorithm 2** End-to-end text receiver pipeline (`Signalverarbeitung_app`)

---

**Require:** `procParam` with fields `rxSignal`, `iNfft`, `iNg`, `iNb`, `iNoTxAnt`, `iNoRxAnt`, `iNoBlocks`, `iNewNoBlocks`, `iNoSubBlocks`, `iModOrd`, `fBBFreq`, `fCarrFreq`, `fs`, `mimoMode`, and text metadata.

**Ensure:** data containing decoded text, BER metrics, and PHY observables.

- 1: `Assert(procParam.DatenTyp is "Text")`
  - 2: Build `params` and `channel` structs from `procParam`
  - 3: `[rxBits, data] = AnalyzeRxSig_app(procParam.rxSignalT, params, channel)`
  - 4: Byte-align `rxBits` and decode ASCII via `bits2text_app`
  - 5: Compute BER versus reference text via `bitFehlerRaten_app`
  - 6: Return data (including `MetricData`, `channelUeb`, `channelImpuls`, `mDataRxEq`)
- 

organized around a single data interface: the App Designer GUI assembles all required parameters and the recorded waveform into `procParam`, and the backend returns one analysis structure (`data`) that serves both payload reconstruction and teaching-oriented visualization.

### 3.4.1 Processing contract and data interface (`procParam` → `data`)

The PHY backend is entered through `Signalverarbeitung_app(procParam)`. In the current build, the payload type is text-only: the function explicitly checks `procParam.DatenTyp == "Text"` and processes the recording accordingly. All parameters needed by the receiver (OFDM dimensions, antenna counts, frequencies, framing metadata) are passed explicitly through `procParam` and mapped into the local structures `params` and `channel`. The receiver output is returned as a single structure `data`, which contains both decoded payload results and intermediate PHY data needed for analysis plots (synchronization metrics, CTF/CIR, equalized symbols).

**Receiver pipeline overview (implementation-level pseudocode).** Algorithm 2 summarizes the executed steps and the concrete function boundaries used in the implementation.

### 3.4.2 Receiver front-end: baseband conversion, timing, and OFDM extraction

The function `AnalyzeRxSig_app` implements a shared OFDM front-end for all supported MIMO modes. Each recorded microphone channel is independently downconverted from passband to complex baseband using `DeModulateSignal_app`. Since resampling and filtering may produce slight length differences across channels, the implementation enforces a common processing length by truncating all channels to the length of the first processed stream.

Timing synchronization is performed per microphone channel using a Schmidl–Cox metric computed by `EstFrameStart_app`. The receiver selects the earliest detected start index across channels and uses it as the global cut point for block extraction. The full metric traces are retained as `MetricData` for GUI inspection.

After timing, the baseband streams are segmented into OFDM blocks of length `iNb=iNfft+iNg` and the cyclic prefix is removed. The resulting array `mFrameRxNoCP` has dimension

---

**Algorithm 3** Front-end synchronization and OFDM block extraction  
**(AnalyzeRxSig\_app)**


---

**Require:**  $\text{rxFrame} [N_r \times N_{\text{samp}}]$ ,  $\text{params} (\text{iNfft}, \text{iNg}, \text{iNb}, \text{fBBFreq}, \text{fDACFreq}, \text{fCarrFreq})$ ,  $\text{iNewNoBlocks}$

**Ensure:**  $\text{mFrameRxNoCP} [N_{\text{FFT}} \times N_{\text{blocks}} \times N_r]$  and  $\text{MetricData}$  per channel

- 1: **for**  $r = 1$  to  $N_r$  **do**
- 2:    $\text{bb}(r, :) = \text{DeModulateSignal\_app}(\text{rxFrame}(r, :), \text{fBBFreq}, \text{fDACFreq}, \text{fCarrFreq})$
- 3: **end for**
- 4: **for**  $r = 1$  to  $N_r$  **do**
- 5:    $[\text{start}(r), \text{cfo}(r), \text{metric}(r)] = \text{EstFrameStart\_app}(\text{bb}(r, :), \text{iNfft}, \text{iNg})$
- 6: **end for**
- 7:  $\text{iFrSync} = \min(\text{start}(\text{isfinite}(\text{start})))$
- 8:  $\text{iFrStart} = \text{clamp}(\text{iFrSync} + \text{iNfft} - \text{safetyMargin})$
- 9: Determine  $\text{iNewNoBlocks}$  from available samples
- 10: Cut  $\text{iNewNoBlocks}$  blocks of length  $\text{iNb}$ , reshape to  $\text{mFrameRx} [iNb \times iNewNoBlocks \times N_r]$
- 11: Remove CP:  $\text{mFrameRxNoCP} = \text{mFrameRx}(\text{iNg}+1:\text{end}, :, :, :)$
- 12: Return  $\text{mFrameRxNoCP}$  and  $\text{MetricData}$

---

$[N_{\text{FFT}} \times N_{\text{blocks}} \times N_r]$  and is passed to the mode-specific receiver function.

**Schmidl–Cox timing and coarse CFO estimate (code-faithful model).** In  $\text{EstFrameStart}\backslash\text{app}$ , the sync symbol has length  $iNfft = 2L$  and consists of two identical halves. For each candidate delay  $d$ , the implementation computes

$$P(d) = \sum_{n=0}^{L-1} r[d+n]^* r[d+n+L], \quad (21)$$

$$R(d) = \sum_{n=0}^{L-1} (|r[d+n]|^2 + |r[d+n+L]|^2), \quad (22)$$

$$M(d) = \frac{|P(d)|^2}{R(d)^2 + \varepsilon}, \quad (23)$$

and selects the frame start as  $d^* = \arg \max_d M(d)$ . The coarse carrier-frequency offset estimate is obtained from the phase of  $P(d^*)$ :

$$\hat{f}_{\text{CFO}} = \frac{\angle P(d^*)}{2\pi L}, \quad (24)$$

where the returned quantity is a normalized offset in cycles per sample, consistent with the implementation.

**Timing front-end (implementation-level pseudocode).** Algorithm 3 reflects the executed control flow in `AnalyzeRxSig_app`.

### 3.4.3 Training structure and per-subframe parsing for spatial multiplexing and V-BLAST

For spatial multiplexing and V-BLAST, the receiver expects a repeated subframe structure in which  $N_t$  dedicated preamble OFDM symbols are placed before each group of `iNoSubBlocks` data symbols. This time-multiplexed sounding design enables a direct per-link channel estimate: during the preamble symbol associated with transmit antenna  $t$ , only that stream carries the known training symbol, so the observed spectrum at each microphone corresponds to a single SISO link.

In `SM_VBLAST_Rx_app`, preambles are extracted into `mPreambleRx` with dimension  $[N_{\text{FFT}} \times N_{\text{subframes}} \times N_r \times N_t]$ , while data blocks are collected into `mDataRx` for subsequent FFT and detection.

### 3.4.4 Training-based channel estimation used in the receiver (ZF/LS via preamble division)

Channel estimation is performed in the frequency domain using a deterministic training spectrum derived from `ChuSeq(iNfft)`. Let  $X_p[k]$  denote the known preamble spectrum on subcarrier  $k$ . Under time-multiplexed sounding, the received preamble on microphone  $r$  when transmit antenna  $t$  is active can be written as

$$Y_{r,t}[k] = H_{r,t}[k] X_p[k] + N_{r,t}[k]. \quad (25)$$

The implemented estimator corresponds to a least-squares / zero-forcing inversion on each subcarrier:

$$\hat{H}_{r,t}[k] = \frac{Y_{r,t}[k]}{X_p[k]}. \quad (26)$$

The resulting channel transfer functions are stored as `channelUeb`. For time-domain inspection, the receiver computes

$$\hat{h}_{r,t}[n] = \sqrt{N_{\text{FFT}}} \text{IFFT}\{\hat{H}_{r,t}[k]\}, \quad (27)$$

and stores the impulse responses as `channelImpuls`. Both representations are visualized in the GUI via the CIR/CTF analysis button.

### 3.4.5 Noise power and SNR proxy from the trailing blocks

To parameterize linear detection, the receiver derives a scalar SNR proxy from a noise-only region. The implementation selects a fixed number of trailing OFDM blocks (up to the last 10) as `mZeroRx` and estimates per-subcarrier noise power by averaging the squared magnitude across these blocks and across microphones. This produces the scalar `iSNR`, which is subsequently used as the regularization strength in MMSE detection.

### 3.4.6 Per-subcarrier MMSE detection and optional decision-directed phase tracking

After FFT of the received data blocks, detection is performed independently on each subcarrier using the estimated MIMO channel. For subcarrier  $k$  in a given subframe, the receiver forms the channel matrix  $\hat{\mathbf{H}}[k] \in \mathbb{C}^{N_r \times N_t}$  from `channelUeb` and applies linear MMSE detection:

$$\hat{\mathbf{x}}[k] = \left( \hat{\mathbf{H}}^H[k] \hat{\mathbf{H}}[k] + \frac{1}{iSNR} \mathbf{I} \right)^{-1} \hat{\mathbf{H}}^H[k] \mathbf{y}[k]. \quad (28)$$

---

**Algorithm 4** Per-subcarrier MMSE detection in spatial multiplexing  
(`SM_VBLAST_Rx_app`)

---

**Require:** `mDataRxFreq` [ $N_r \times N_{\text{FFT}} \times N_{\text{blocks}}$ ], `mCTF` [ $N_{\text{FFT}} \times N_{\text{subframes}} \times N_r \times N_t$ ], scalar `iSNR`

**Ensure:** `mDataRxEq` [ $N_t \times N_{\text{FFT}} \times N_{\text{blocks}}$ ]

```

1: for each subframe  $s$  do
2:   for each data block  $b$  in subframe  $s$  do
3:     for each subcarrier  $k = 1 \dots N_{\text{FFT}}$  do
4:        $\hat{\mathbf{H}} \leftarrow \text{squeeze}(\text{mCTF}(k, s, :, :))$   $\triangleright [N_r \times N_t]$ 
5:        $\mathbf{A} \leftarrow \hat{\mathbf{H}}^H \hat{\mathbf{H}} + \mathbf{I} \cdot (1/\text{iSNR})$ 
6:        $\hat{\mathbf{x}} \leftarrow \sqrt{N_t} \mathbf{A}^{-1} \hat{\mathbf{H}}^H \mathbf{y}$   $\triangleright \mathbf{y} = \text{mDataRxFreq}(:, k, b)$ 
7:       mDataRxEq(:, k, b)  $\leftarrow \hat{\mathbf{x}}$ 
8:     end for
9:   end for
10: end for

```

---

The equalized symbol tensor is stored as `mDataRxEq` with dimension  $[N_t \times N_{\text{FFT}} \times N_{\text{blocks}}]$  and directly feeds constellation/EVM visualization.

To stabilize residual common phase rotation, an optional decision-directed phase tracker is applied per transmit stream. Hard symbol decisions are used to construct reference symbols, and a phase estimate is obtained from the argument of the correlation sum. The estimate is smoothed by a first-order recursion and compensated on the equalized symbols.

**Subcarrier-wise MMSE detection (implementation-level pseudocode).** Algorithm 4 captures the core loop in `SM_VBLAST_Rx_app` for the spatial multiplexing path.

### 3.4.7 Symbol demapping and ASCII reconstruction

Finally, the receiver serializes the detected symbols and recovers bits in a text-specific format: the first `len_cInfoBits` positions are demapped as BPSK header bits, and the remaining positions are demapped as  $M$ -QAM payload bits using the configured modulation order `iModOrd`. The bitstream is byte-aligned and converted to ASCII in `bits2text_app`. BER is computed by comparing the decoded text against the original reference payload passed to the receiver.

### 3.4.8 Stored observables for GUI-based analysis

Throughout the chain, the receiver retains intermediate observables in the analysis structure returned to the GUI. In particular, synchronization traces (`MetricData`), channel estimates (CTF/CIR), and equalized symbols are stored explicitly. This design allows the GUI to provide targeted analysis views (synchronization plots, CIR/CTF inspection, rank-related measures, constellation and EVM) without rerunning the full processing chain.

### 3.5 Chapter summary

This chapter has described how the demonstrator is realized as a reproducible end-to-end acoustic MIMO–OFDM link: a structured GUI-to-PHY interface, a shared OFDM front-end, and mode-specific MIMO processing stages with training-based channel estimation and per-subcarrier detection. Beyond recovering the text payload, the implementation deliberately preserves intermediate results—synchronization traces, channel responses, equalized symbols, and derived quality indicators—so that each processing decision remains explainable and visually inspectable.

These stored observables form the basis of the experimental chapter that follows. Chapter 4 reports representative measurements obtained with the implemented system and presents the corresponding plots (synchronization metrics, CIR/CTF, constellation/EVM, and rank-related measures) without interpretation, establishing a clean separation between *how the system is built* (this chapter) and *what it produces* (next chapter).

## 4 Results

The implementation described in the previous chapters results in a stable and repeatedly executable acoustic MIMO demonstrator that supports end-to-end transmission of text payloads over a multi-channel link. Besides successful payload recovery, the system exposes a set of physical-layer observables that make the reception process inspectable: synchronization metrics, frequency- and time-domain channel estimates (CTF/CIR), equalized symbol streams, rank-related measures, and error vector magnitude (EVM). The following sections present these results in a descriptive manner, leaving interpretation and discussion to the subsequent chapter.

Unless stated otherwise, the results presented in this chapter correspond to the **main operating mode** of the demonstrator, which is configured as follows:

- Transmission mode: **Spatial Multiplexing (SM)**
- Acoustic array configuration: **4 loudspeakers (Tx) and 8 microphones (Rx)**
- Modulation scheme: **4-QAM** ( $M = 4$ , corresponding to `iMod0rd = 2`)
- Channel estimation: **Zero-Forcing (ZF)** estimator
- Equalization: **Minimum Mean Square Error (MMSE)** equalizer
- Channel coding: **disabled**

### 4.1 Recorded transmit and receive signals

The demonstrator provides direct access to the recorded acoustic transmit and receive signals, enabling inspection of the waveform content at different signal representations. For each microphone channel, the recorded receive signal can be examined over the full recording interval, and—when the corresponding transmit sequence is available—the transmitted loudspeaker signals can be inspected for reference.

In addition to the time-domain representation, the system exposes frequency-domain magnitude spectra for all channels, allowing the occupied bandwidth and channel-dependent spectral shaping introduced by the acoustic path and the audio front-end to be observed.

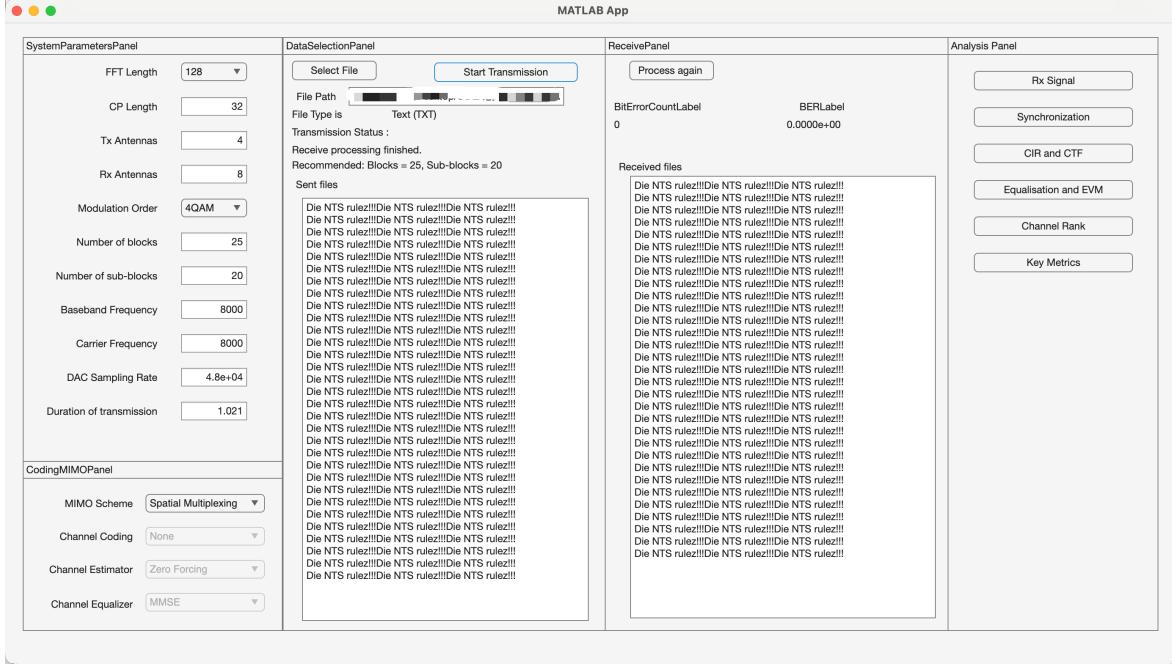


Figure 3: Default UI main interface for 4T8R modulated via 4QAM in SM mode

Furthermore, a baseband spectral view centered around the configured carrier frequency is provided by extracting and re-centering the spectral region of interest. This compact representation highlights the effective transmission band used by the MIMO-OFDM signal and serves as a consistency check for modulation, upconversion, and recording.

## 4.2 Timing synchronization results

Frame timing synchronization is evaluated using a Schmidl–Cox-style repeated training structure. Figure 4 summarizes the synchronization observables obtained for the main operating mode across all eight receive channels (microphones).

For each microphone, three quantities are reported as functions of the delay index  $d$ : the magnitude of the correlation sum  $|P(d)|$ , the corresponding energy term  $R(d)$ , and the normalized timing metric  $M(d)$ . These quantities are shown in a consistent three-row layout, with columns corresponding to individual receive channels.

Across all microphones, a dominant peak is observed at a common delay index. This peak is clearly visible in  $|P(d)|$  and  $R(d)$ , and results in a pronounced maximum in the timing metric  $M(d)$ . The alignment of the peak position across all receive channels indicates that a consistent frame start can be detected despite channel-dependent amplitude variations.

While the absolute magnitudes of  $|P(d)|$  and  $R(d)$  differ between microphones, reflecting different received signal powers and acoustic channel conditions, the location of the synchronization peak remains stable. In the implementation, the detected peak index (`idxMax`) is extracted per channel and used to derive a common frame start for subsequent OFDM block extraction and MIMO processing.

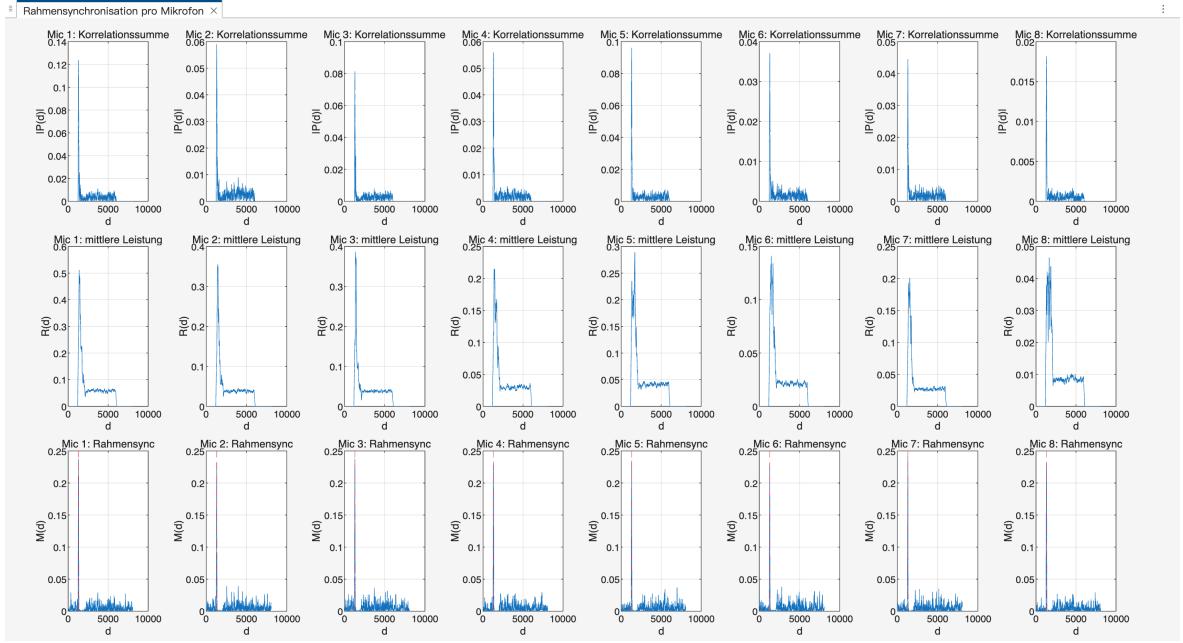


Figure 4: Synchronization observables per receive channel for the main operating mode (SM, 4-QAM, 4 transmit loudspeakers and 8 receive microphones). Top row: correlation magnitude  $|P(d)|$ ; middle row: energy term  $R(d)$ ; bottom row: normalized timing metric  $M(d)$ .

### 4.3 Channel estimation outputs: CIR and CTF

After frame synchronization and cyclic-prefix removal, the receiver estimates the frequency-domain channel transfer function (CTF) from the preamble symbols. The corresponding time-domain channel impulse response (CIR) is then obtained via an inverse FFT. For each transmit–receive channel pair, both representations are evaluated for the same selected subframe and visualized jointly.

Figure 5 presents the estimated channel responses for the main operating mode. For a fixed transmit channel, the results are shown across all eight receive microphones. The upper row displays the magnitude of the CIR,  $|h[n]|$ , as a function of the delay tap index, while the lower row shows the magnitude of the CTF,  $|H[k]|$ , across OFDM subcarriers on a logarithmic (dB) scale.

In the CIR plots, a small number of dominant taps can be observed at low delay indices for all receive channels, followed by a rapidly decaying tail. The relative strength and exact tap distribution vary across microphones, reflecting channel-dependent propagation conditions. For visualization purposes, the CIRs are normalized to enable direct comparison between different receive channels.

The corresponding CTF plots exhibit pronounced frequency selectivity, with deep notches and peaks that differ between receive channels. These spectral variations are consistent with the multipath structure observed in the CIR representations and illustrate the frequency-dependent nature of the acoustic MIMO channel. Together, the paired CIR and CTF views provide complementary insight into the same estimated channel realization in the time and frequency domains.

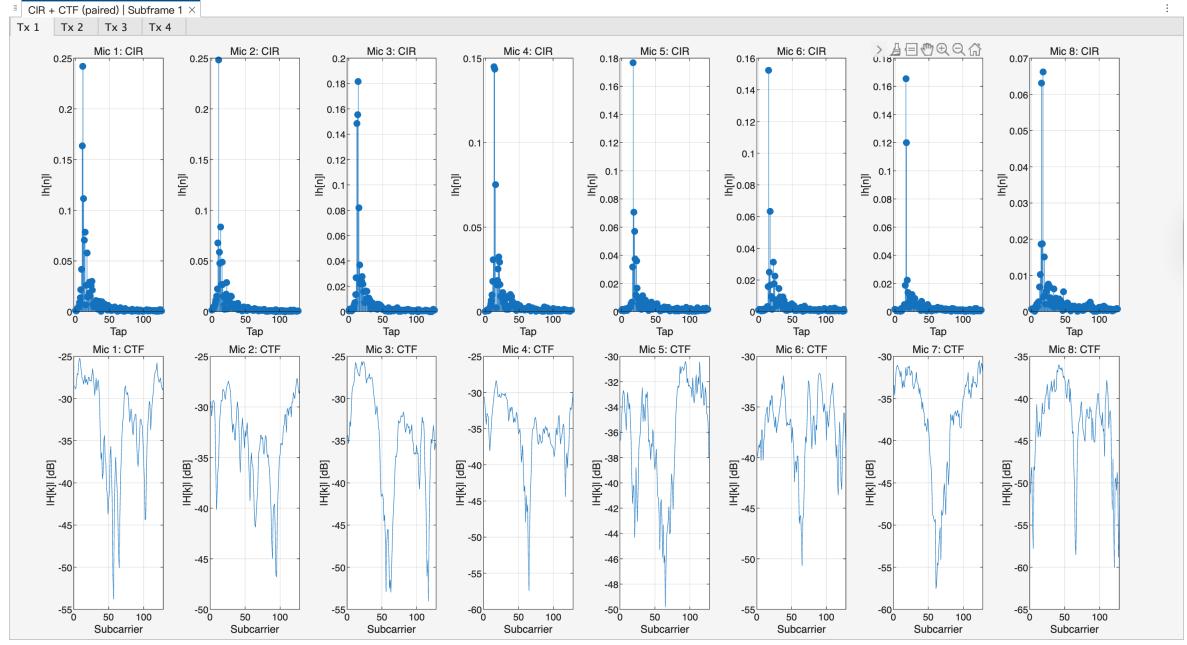


Figure 5: Estimated channel representations for the main operating mode (SM, 4-QAM, 4 transmit loudspeakers and 8 receive microphones). Top row: magnitude of the channel impulse response  $|h[n]|$ ; bottom row: magnitude of the channel transfer function  $|H[k]|$  in dB. Results are shown for the same selected subframe and grouped by transmit channel.

#### 4.4 Equalized symbol-domain results: constellations and EVM

After MIMO detection and linear equalization, the receiver provides access to the equalized complex-valued symbols for each spatial stream. For the main operating mode (SM, four transmit loudspeakers and eight receive microphones, 4-QAM), the results are organized per transmit stream and evaluated in both the complex symbol domain (constellation plots) and the error domain using EVM-based metrics. All results reported in this section are derived from payload symbols only; symbol positions associated with header or control information are explicitly excluded from the EVM computation.

##### 4.4.1 Equalized constellations per spatial stream

Figure 6 shows the equalized symbol constellations for the four spatial streams corresponding to the four transmit loudspeakers. Each constellation aggregates symbols across all active subcarriers and OFDM blocks of the selected frame.

The four QAM clusters are clearly identifiable in all streams, indicating successful separation of the spatial layers by the receiver. At the same time, the apparent spread of the clusters differs between streams, reflecting stream-dependent channel and noise conditions after equalization. The constellations are shown without any post-processing such as decision-directed refinement, so the observed dispersion directly corresponds to the output of the linear MMSE equalizer.

##### 4.4.2 EVM-based quality metrics

To quantify the quality of the equalized symbols, the error vector magnitude (EVM) is evaluated with respect to the nearest ideal 4-QAM constellation points. Figure 7 summarizes the EVM results in three complementary representations.

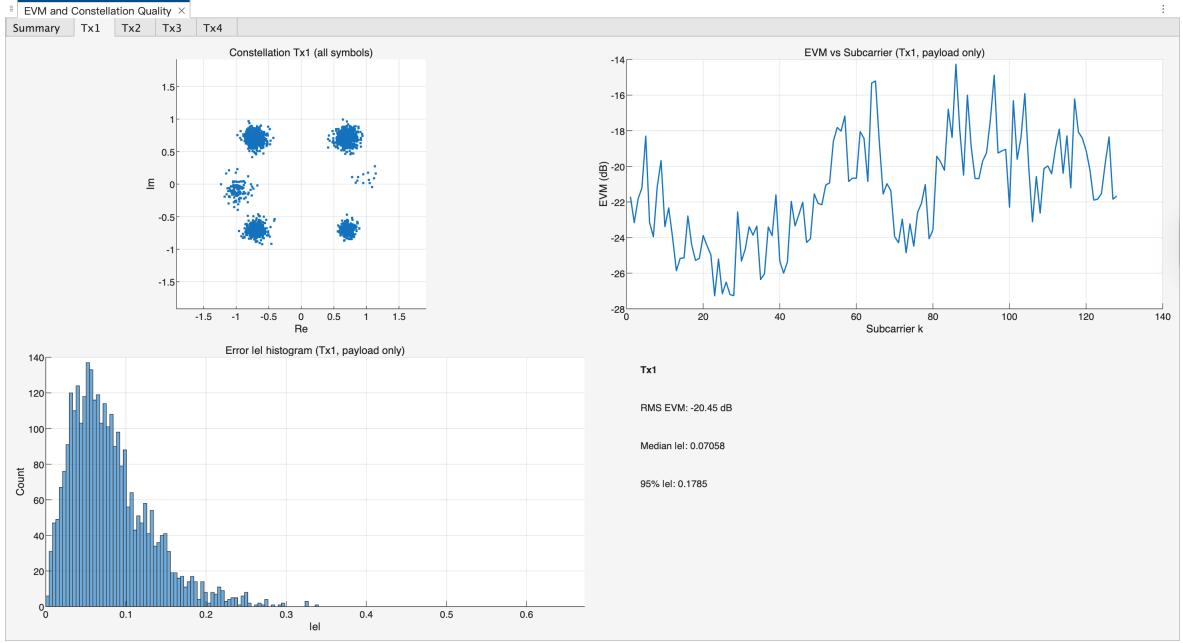


Figure 6: Equalized symbol constellations per spatial stream for the main configuration (SM, four transmit loudspeakers, eight receive microphones, 4-QAM).

First, the RMS EVM per spatial stream provides a compact scalar measure of symbol quality after equalization. In the shown example, the four streams exhibit different RMS EVM levels, consistent with the stream-dependent dispersion observed in the constellation plots.

Second, the EVM is reported as a function of subcarrier index for each stream. This representation highlights frequency-selective variations across the OFDM band and allows spatial streams to be compared on a per-subcarrier basis.

Third, histograms of the error magnitude  $|e|$  illustrate the statistical distribution of symbol errors for each stream. These histograms provide additional insight into the spread and tail behavior of the error distribution beyond a single RMS value.

## 4.5 Channel rank and conditioning results

To characterize the spatial properties of the estimated acoustic MIMO channel, the demonstrator evaluates rank- and conditioning-related metrics of the frequency-domain channel matrix  $H[k]$  for each subcarrier. Figure 8 summarizes these results for the main operating mode.

### 4.5.1 Rank over subcarriers and rank histogram

The numerical rank of the channel matrix  $H[k] \in \mathbb{C}^{N_r \times N_t}$  is computed for each subcarrier using a singular-value-based criterion with a fixed tolerance. The top-left panel of Figure 8 shows the rank as a function of the subcarrier index.

For the considered configuration with  $N_t = 4$  transmit channels and  $N_r = 8$  receive channels, the rank is equal to four for all subcarriers. This is further confirmed by the rank histogram shown in the top-right panel, where all observations fall into a single bin centered at rank four. No rank deficiency is observed across the evaluated bandwidth.

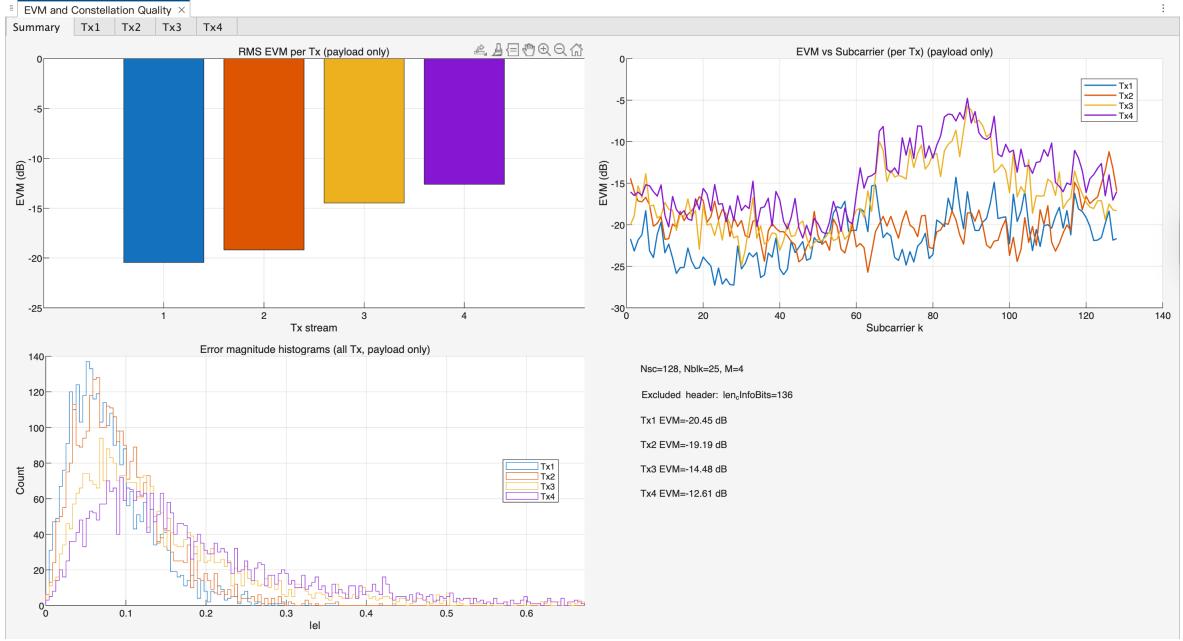


Figure 7: EVM evaluation for the main configuration (SM, four transmit loudspeakers, eight receive microphones, 4-QAM): RMS EVM per stream, EVM versus subcarrier index, and error-magnitude histograms (payload symbols only).

#### 4.5.2 Minimum singular value and condition-number proxy

In addition to rank, two auxiliary measures are reported to assess the numerical conditioning of the channel matrices. The bottom-left panel of Figure 8 shows the minimum singular value  $\sigma_{\min}(H[k])$  as a function of subcarrier index, expressed in dB. While  $\sigma_{\min}(H[k])$  varies across frequency, it remains finite for all subcarriers, consistent with the full-rank observation.

The bottom-right panel shows a condition-number proxy defined as

$$\text{cond}(H[k]) = \frac{\sigma_{\max}(H[k])}{\sigma_{\min}(H[k])},$$

plotted on a logarithmic scale. The condition number exhibits frequency-dependent variations, indicating that although the channel remains full rank, the relative separation between singular values changes across subcarriers.

Together, these plots provide a compact summary of the spatial degrees of freedom and numerical conditioning of the estimated MIMO channel over frequency.

## 4.6 End-to-end payload reconstruction and bit error results

Besides intermediate PHY observables, the demonstrator also provides an end-to-end view of whether the payload can be reconstructed successfully. After demapping, the recovered payload is presented in the GUI (text preview area), and—if the original transmitted payload is available as reference—a bitwise comparison is performed to obtain the bit error count and the corresponding BER. In the main configuration reported in this chapter (SM, 4QAM, no channel coding), these indicators reflect the uncoded link performance.

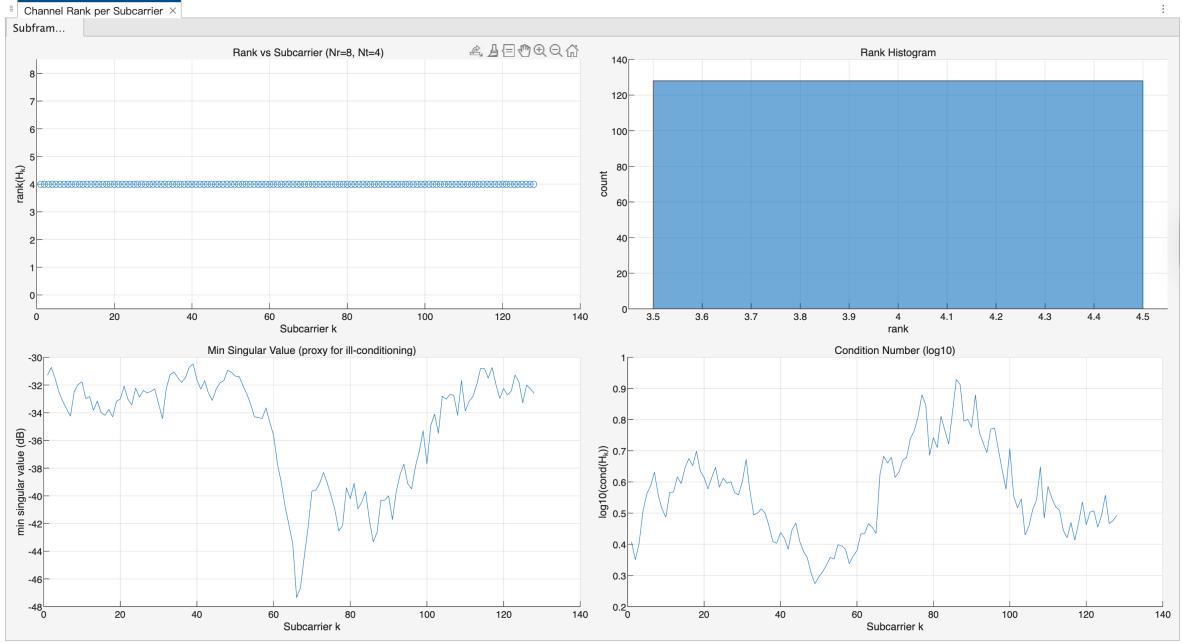


Figure 8: Channel rank and conditioning measures for the main operating mode. Top: rank versus subcarrier index and corresponding rank histogram. Bottom: minimum singular value (in dB) and condition-number proxy versus subcarrier.

#### 4.6.1 Recovered text output

In text mode, the reconstructed payload is formed by converting the recovered bitstream back to ASCII characters. In the current implementation, this includes (i) optional removal of control/header bits when such a header is configured, (ii) truncation to the known payload length when available, (iii) enforcing byte alignment, and (iv) mapping bytes to characters. The decoded string is displayed directly in the GUI after receiver processing, providing immediate qualitative confirmation of successful reconstruction.

#### 4.6.2 Bit error count and BER (GUI indicators)

When a reference payload is available (i.e., the transmitted text is stored alongside the run), the demonstrator computes the bit error count and BER using `bitFehlerRaten_app`. For text payloads, the function converts both the transmitted and reconstructed strings to 8-bit ASCII sequences via `de2bi(.,8)`, reshapes them into a linear bitstream, and then applies `biterr` on the common prefix length:

$$\text{BER} = \frac{N_{\text{err}}}{N_{\text{comp}}}, \quad N_{\text{comp}} = \min\{N_{\text{tx}}, N_{\text{rx}}\}.$$

The resulting values are presented in the GUI via two numeric indicators (bit error count and BER). Figure 9 shows an example of this UI output for a representative run in which the displayed values are zero.

*Implementation note (to be reconciled):* Channel coding is currently not enabled in the receiver chain. While a coding mode parameter exists on the transmitter side in the broader project, the reported BER is therefore an uncoded BER. A coded BER evaluation requires consistent integration of receiver-side channel decoding.

BitErrorCountLabel	BERLabel
0	0.0000e+00

Figure 9: GUI indicators for end-to-end bit error evaluation: bit error count and BER for one representative run.

Table 5: Placeholder for the key-metrics snapshot exported from the demonstrator for one representative run.

Metric	Value
$N_{\text{FFT}}$	(to be inserted)
$N_g$ (CP length)	(to be inserted)
Modulation ( $M$ )	(to be inserted)
$N_{\text{T}} / N_{\text{R}}$	(to be inserted)
Frame start used	(to be inserted)
$i\text{SNR}$ (Average SNR)	(to be inserted)
EVM per stream (if stored)	(to be inserted)
Rank mean/min/max (if stored)	(to be inserted)

## 4.7 Key metrics summary (tool-reported snapshot)

For rapid inspection and teaching-oriented demonstrations, the GUI provides a consolidated “Key Metrics” view that lists selected parameters and computed metrics from the receiver processing. Typical entries include FFT and CP sizes, modulation order, the number of transmit/receive channels, the selected frame start index, and (when stored) SNR-related quantities, EVM per stream, and rank summary statistics.

*Implementation note (to be reconciled):* Some metrics are mode-dependent and may be stored under different field names or only in specific receiver branches. The “Key Metrics” view therefore reports available quantities for the selected mode and run.

## 4.8 From results to discussion

This chapter has presented the set of observable outputs produced by the MIMO audio demonstrator under its main operating configuration. These results establish a concrete basis for further analysis, as they expose the behavior of synchronization, channel estimation, spatial characteristics, and symbol-level quality in a real acoustic MIMO setting. In the following chapter, these observations are examined in more detail and compared across operating modes to discuss their implications for system performance and robustness.

# 5 Discussion

Chapter 4 presented the observable results obtained with the implemented acoustic MIMO–OFDM demonstrator, including constellation diagrams, channel impulse and frequency responses, synchronization metrics, and end-to-end decoding performance. The results demonstrate that reliable text transmission is achievable under practical acoustic conditions, while at the same time revealing several non-ideal effects at the symbol and channel level.

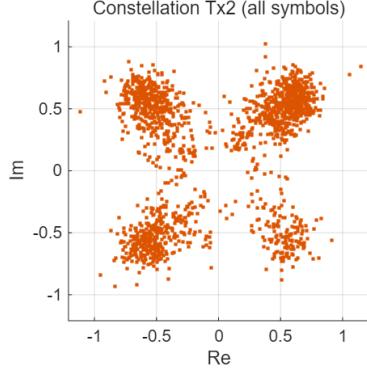


Figure 10: Short cyclic prefix ( $N_{\text{FFT}} = 128$ ,  $N_g = 32$ ): dispersed constellation cloud despite reliable decoding ( $\text{BER} \approx 0$ ).

The purpose of this chapter is not to benchmark theoretical limits, but to interpret the observed phenomena in the context of the *actual implementation*. Each discussion point is therefore explicitly linked to the receiver algorithms, parameter choices, and synchronization strategy used in the system.

## 5.1 Reliable decoding despite dispersed constellations

A first striking observation is shown in Fig. 10. Although the constellation exhibits pronounced spreading and appears as a “cloud” rather than compact clusters, the measured bit error rate is zero. This behavior is not accidental, but a direct consequence of the modulation, detection, and equalization strategy used in the demonstrator.

The system operates with uncoded 4-QAM modulation and hard-decision nearest-neighbor demapping. As long as the equalized symbols remain within their correct decision regions, even substantial amplitude and phase distortion does not necessarily lead to bit errors. Consequently, BER may remain zero even when constellation quality is clearly non-ideal.

This effect is reinforced by MMSE equalization. Unlike zero-forcing, MMSE explicitly trades residual interference for reduced noise enhancement. As a result, equalized symbols are not forced onto ideal constellation points, which manifests as increased dispersion in the constellation even when the hard decisions remain correct.

## 5.2 Influence of cyclic prefix length on constellation quality

A clear improvement in constellation compactness is observed when the cyclic prefix length is increased from 32 to 256 samples, as shown in Fig. 11. This effect is consistent with the measured channel impulse responses: in several microphone channels, the effective multipath tail extends beyond 32 samples. When the cyclic prefix is shorter than the channel delay spread, residual inter-symbol interference leaks into the FFT window, violating the circular convolution assumption of OFDM and introducing subcarrier-dependent distortion.

With a longer cyclic prefix, the dominant part of the impulse response is contained inside the prefix, and the OFDM demodulation assumptions are restored. The result is a noticeably tighter constellation, even without changing the equalizer or estimator.

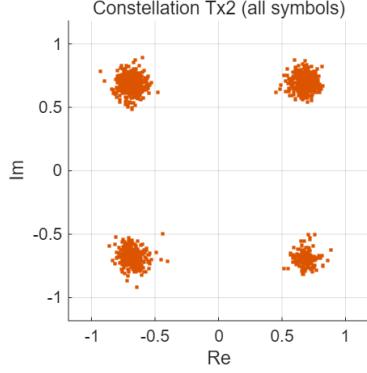


Figure 11: Long cyclic prefix ( $N_{\text{FFT}} = 128$ ,  $N_g = 256$ ) at nominal volume: constellation clusters become significantly more compact, indicating reduced ISI.

### 5.3 Global frame start selection and CIR structure

Figures 12 and 13 highlight the impact of the global frame-start selection strategy on the estimated channel impulse responses (CIR). In the current implementation, synchronization is performed independently on each receive channel using the Schmidl–Cox metric (`EstFrameStart_app`). The receiver then selects the earliest detected peak across microphones as a *global* reference start index, followed by a fixed safety margin:

$$i_{\text{start}} = \min_i \{\hat{i}_i\} + N_{\text{FFT}} - \text{safetyMargin}.$$

In the provided code, `safetyMargin = 10`. This choice favors robust global operation: shifting slightly earlier reduces the risk that noisy peak differences across microphones cause the FFT window to start too late and cut into useful signal energy.

The trade-off is visible in the CIR tails. With `safetyMargin=10` (Fig. 12), some channels show small residual components toward the end of the impulse response. These are consistent with window misalignment artifacts caused by using a single global FFT window for all microphones rather than optimizing the window per channel.

When `safetyMargin=0` (Fig. 13), the CIR tails of some channels appear cleaner and decay more smoothly. However, this comes with reduced robustness against synchronization uncertainty: even small timing errors can shift the FFT window into a less favorable region, which may increase distortion in other runs or environments.

### 5.4 Constellation contraction at low transmit amplitude

Figure 14 shows the constellation obtained when the transmit amplitude is reduced while using a long cyclic prefix. The symbol clusters clearly contract toward the origin.

This effect should not be attributed to carrier-frequency offset. Because transmit and receive run on the same multi-channel audio interface, both sides share a common hardware clock; thus, true CFO is expected to be negligible. In this operating regime, the more plausible explanation is reduced effective SNR combined with MMSE equalization. As the transmit amplitude is lowered, the noise contribution becomes more dominant. The MMSE equalizer then increasingly regularizes the inversion of the channel, resulting in an effective gain reduction that pulls equalized symbols toward the origin.

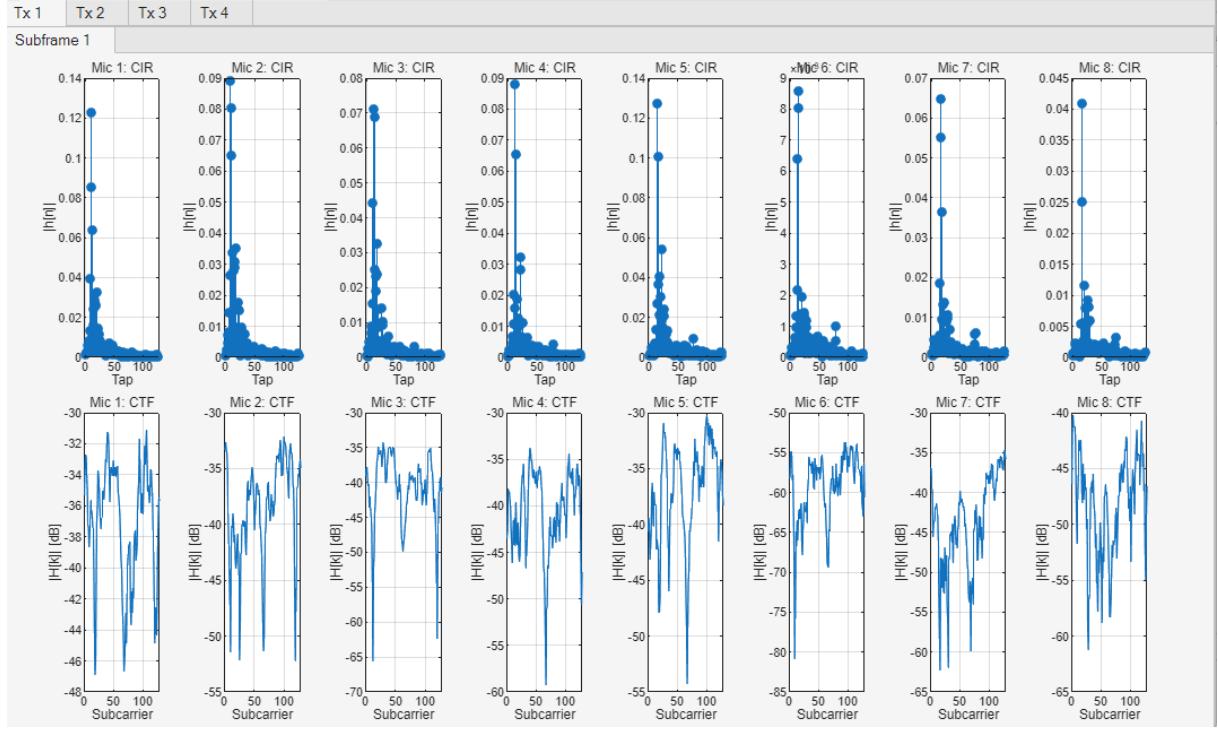


Figure 12: Estimated CIR/CTF with `safetyMargin=10`. The global windowing is robust, but small residual tail components appear in some channels.

## 5.5 Engineering implications and refinement options

The observed non-idealities are consistent with the implementation choices of the demonstrator, which prioritizes robust and reproducible operation over per-channel optimality. The results also indicate clear refinement options that would mainly improve symbol-domain interpretability:

- **Per-channel fine timing after global start:** keep the global start for consistency, but apply a small per-microphone timing refinement (within a limited search window) to maximize CIR energy concentration inside the cyclic prefix.
- **Adaptive safety margin:** replace the fixed `safetyMargin` by an adaptive value derived from the observed metric peak width or from CIR energy distribution.
- **Post-equalization gain normalization:** apply a stream-wise complex gain correction after MMSE equalization to reduce amplitude contraction and improve constellation comparability across runs.

These adjustments would strengthen the didactic value of constellation/CIR/CTF visualizations while maintaining the stable end-to-end behavior demonstrated in Chapter 4.

## 6 Conclusion

This thesis addressed the development of a MIMO-OFDM audio demonstrator with the dual purpose of enabling hands-on experimentation and supporting teaching-oriented

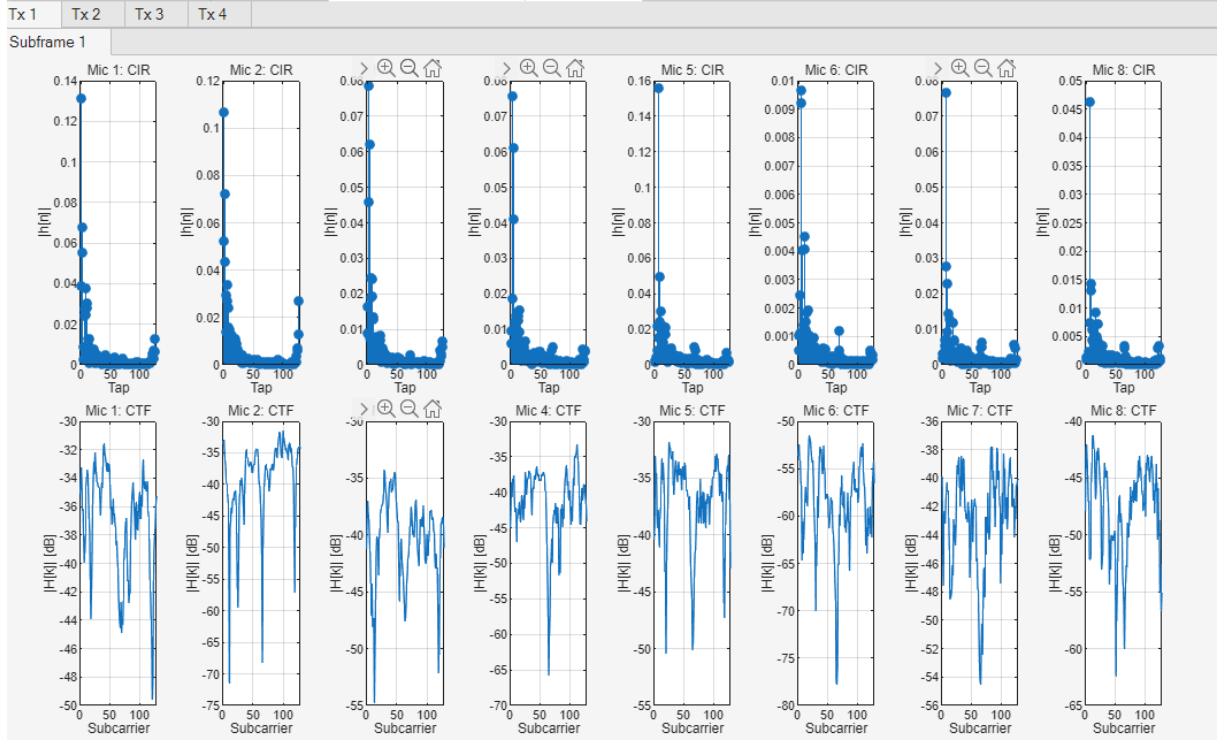


Figure 13: Estimated CIR/CTF with `safetyMargin=0`. CIR tails appear cleaner for some channels, but robustness against timing uncertainty is reduced.

analysis of modern physical-layer concepts. Starting from an existing software framework, the work focused on adapting the system to new hardware constraints, restructuring the user interface, and extending the receiver processing chain to expose meaningful and interpretable intermediate results.

## 6.1 Summary of achieved objectives

The thesis objectives defined in Chapter 1 have been met as follows.

**Objective 1: Adaptation to a multi-channel audio interface.** The demonstrator was successfully adapted to operate with a modern multi-channel audio interface, supporting simultaneous playback and recording across multiple loudspeaker and microphone channels. A stable streaming setup was achieved using frame-based audio I/O, with explicit handling of buffering, latency, and synchronization constraints. The resulting system enables repeatable transmission and reception of broadband OFDM waveforms in an acoustic environment, forming a reliable experimental basis for further signal processing and analysis.

**Objective 2: Migration and restructuring of the graphical user interface.** The legacy GUIDE-based GUI was migrated to MATLAB App Designer and restructured to clearly separate user interaction from signal-processing logic. The resulting application provides intuitive configuration of physical-layer parameters, mode selection, and execution control, while exposing a rich set of visualization and analysis tools. This modular

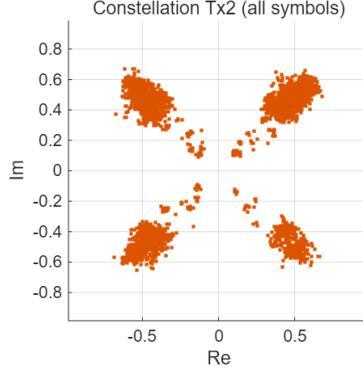


Figure 14: Low transmit amplitude ( $N_{\text{FFT}} = 128$ ,  $N_g = 256$ ): constellation clusters contract toward the origin, consistent with reduced SNR and MMSE regularization.

design improves maintainability and extensibility, and aligns the software structure with current MATLAB development practices.

#### **Objective 3: Implementation and analysis of a MIMO-OFDM receiver chain.**

A configurable receiver processing pipeline was implemented, including timing synchronization, carrier-frequency-offset compensation, channel estimation, MIMO equalization/detection, symbol demapping, and payload reconstruction. The receiver supports spatial multiplexing operation with multiple estimation and equalization strategies and exposes intermediate quantities such as synchronization metrics, CIR/CTF estimates, equalized symbols, channel rank, and EVM. These outputs allow systematic evaluation of MIMO-OFDM behavior and form the core analytical contribution of the demonstrator.

## 6.2 Key outcomes and insights

The experimental results demonstrate that the implemented system operates stably in its main configuration (spatial multiplexing, 4T8R, 4-QAM, zero-forcing channel estimation, MMSE equalization, no channel coding) and reliably supports end-to-end text transmission. Beyond payload recovery, the system provides a coherent set of physical-layer observables whose behavior is consistent with theoretical expectations for frequency-selective MIMO channels.

In particular, the combination of channel estimation, rank analysis, and symbol-domain quality metrics illustrates how spatial multiplexing performance varies across subcarriers and spatial streams. The availability of such intermediate results is especially valuable in an educational context, where understanding system behavior is often more important than optimizing absolute performance figures.

## 6.3 Consolidation and future perspectives

From an engineering perspective, the most relevant outcome of this thesis is not the optimization of a single transmission configuration, but the creation of a *structured, extensible, and transparent* MIMO-audio demonstrator. The current implementation intentionally focuses on a stable baseline configuration and on exposing intermediate physical-layer observables in a form that is easy to inspect, visualize, and reason about.

Some limitations of the present system naturally follow from this design focus. Advanced features such as channel coding, alternative estimation or equalization strategies,

and extended performance metrics are not emphasized in the current evaluation. These aspects are not treated as deficiencies of the system, but rather as conscious design boundaries chosen to preserve clarity, robustness, and didactic value.

At the same time, the modular signal-processing pipeline, the centralized data management, and the GUI-based visualization framework provide a solid foundation for future extensions. The clear separation between transmission, reception, analysis, and visualization enables new functionality to be added with minimal structural changes. Examples include the integration of additional MIMO modes, alternative channel estimators and equalizers, larger or more diverse payload types, and more advanced interaction concepts.

Beyond algorithmic extensions, the developed user interface plays a central role in enabling future experimentation. By offering direct access to key parameters and by visualizing intermediate results in a consistent manner, the demonstrator lowers the barrier for iterative development, comparative studies, and exploratory “what-if” experiments. This makes the system well suited not only for further technical enhancement, but also as a reusable platform in laboratory exercises and project-based teaching.

**Concluding remark.** In summary, this thesis demonstrates that a carefully structured software design, combined with a transparent user interface and a reproducible experimental workflow, can transform an audio-based MIMO-OFDM system into a versatile exploration platform. Rather than aiming for maximal performance, the developed demonstrator prioritizes insight, extensibility, and usability—thereby providing a practical bridge between communication theory and hands-on experimentation, and a flexible starting point for future development.

## References

- [1] R. v. Nee and R. Prasad, *OFDM for Wireless Multimedia Communications*. Artech House, 2000.
- [2] J. G. Proakis and M. Salehi, *Digital Communications*. McGraw-Hill, 5 ed., 2001.
- [3] E. Telatar, “Capacity of multi-antenna gaussian channels,” *European Transactions on Telecommunications*, vol. 10, no. 6, pp. 585–595, 1999.
- [4] G. J. Foschini, “On limits of wireless communications in a fading environment when using multiple antennas,” *Wireless Personal Communications*, vol. 6, no. 3, pp. 311–335, 1998.
- [5] E. G. Larsson, O. Edfors, F. Tufvesson, and T. L. Marzetta, “Massive mimo for next generation wireless systems,” *IEEE Communications Magazine*, vol. 52, no. 2, pp. 186–195, 2014.
- [6] M. Stojanovic, “On the relationship between capacity and distance in an underwater acoustic communication channel,” *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 11, no. 4, pp. 34–43, 2007.
- [7] B. Li, S. Zhou, M. Stojanovic, L. Freitag, and P. Willett, “Mimo-ofdm for high-rate underwater acoustic communications,” *IEEE Journal of Oceanic Engineering*, vol. 34, no. 4, pp. 634–644, 2009.
- [8] M. Stojanovic, “Ofdm for underwater acoustic communications: Adaptive synchronization and sparse channel estimation,” *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008.
- [9] L. Häring, “Ofdm – orthogonal frequency division multiplexing,” 2025. Lecture slides, Chair of Communication Systems, University of Duisburg-Essen.
- [10] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge University Press, 2005.
- [11] “audioplayerrecorder.” MathWorks Documentation. Accessed 2026-01-22.
- [12] “Audio i/o: Buffering, latency, and throughput.” MathWorks Documentation. Accessed 2026-01-22.
- [13] “guide (removed) create or edit ui file in guide.” MathWorks Documentation. Accessed 2026-01-22.
- [14] “Goodbye guide, hello app designer: Evolving your matlab apps.” MathWorks Blog, 2025. Accessed 2026-01-22.