Universität Rostock

Traditio et Innovatio

Digital Communications
Digitale Datenübertragung
Project

– SS 2023 –

Universität Rostock
Institut für Nachrichtentechnik
Prof. Dr.-Ing. Volker Kühn
Albert-Einstein-Strasse 26
18059 Rostock

http://www.int.uni-rostock.de

Daniel Franz
Room 306
Tel.: 0381/498-7314
daniel.franz@uni-rostock.de

Version from
April 18, 2023

Institut für Nachrichtentechnik
Fakultät für Informatik und Elektrotechnik

General Information

- **Do not hesitate to ask questions!!!** (German or English)
- Conceptually, the project is based on a hardware-in-the-loop concept. While the physical transmission is performed using a dedicated communication hardware, transmitter and receiver algorithms are implemented in software running on a host PC. For this you have to implement an Orthorgonal Frequency Division Multiplexing (OFDM) system in MATLAB. You can form groups of two to three students and solve the tasks together.
- Since you will have to write your own scripts and functions in MATLAB you should have at least some basic knowledge. We provide a short introduction to MATLAB in the first two exercises. If this is not sufficient for you, we insist you to use the Mathworks tutorial or the tons of MATLAB books available at the library to improve you MATLAB skills.
- Be aware that your MATLAB code will be discussed in the final examination. Without your own code, which you have to provide regularly, you will hardly pass the exam.
- You can download all necessary files from Stud.IP. Section 1 provides a brief review about the theoretical background. In Section 2 the Lyrtech Hardware is introduced. Further details about your tasks are given in Section 3. If you have any questions about the general organization, feel free to ask your supervisor.
- For students Matlab is available at the Remote-Desktop-Server of the University of Rostock. If you use Windows, run `mstsc.exe` and connect to `unicomp.uni-rostock.de`. If your studentID is `xx001`, your login name should be `rechenzentrum\xx001`. See also `https://www.itmz.uni-rostock.de/en/internet-services/application-server-itmz/application-server-itmz/`.
  You might also download Matlab from the Software Server of the University of Rostock `https://www.itmz.uni-rostock.de/en/applications/software/purchase/informationen-ueber-bestehende-vertraege/mathworks/` for your own PC. Due to license constraints, MATLAB only runs inside the campus network. If you are not in the campus network, you need to connect to it via VPN.

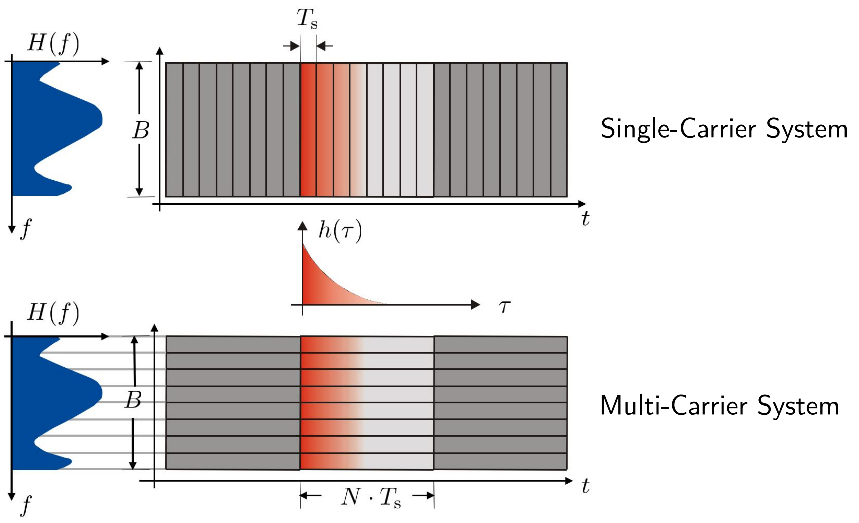

Figure 1: Philosophy of Multi-Carrier Communications

1 Theoretical Review

The main focus on this lecture lies on frequency selective channels. A channel behaves frequency selective if its amplitude response varies significantly within signal bandwidth. In this case the impulse response is not a single Dirac impulse, which leads to intersymbol interference (ISI). During the lecture you learn different techniques to deal with ISI. An optimal detection strategy is the Viterbi algorithm. However, the complexity of this algorithm grows exponentially by length L_h of the channel impulse response. Since increasing data rates also means an increasing impulse response length, this detection strategy becomes highly suboptimal.

A completely different approach for a robust transmission over ISI channels is Orthogonal Frequency Division Multiplexing (OFDM). The general idea is depicted in Figure 1. The bandwidth B in which the channel is frequency selective is divided into multiple subcarrier. Each subcarrier has a smaller bandwidth in which the channel is approximately not frequency selective anymore. In this project, data transmission over a frequency selective channel shall be realized using OFDM.

Discrete-time OFDM system model The complete discrete-time OFDM system model is given in Figure 2. On transmitter side a data stream to be sent is divided into N parallel data streams, which are mapped to symbols. The streams are transmitted on orthogonal subcarriers to avoid intercarrier interference. This is efficiently implemented in a blockwise fashion using the IFFT. It transforms the data from the frequency domain to the time domain. A guard interval has to be added to each IFFT output vector in order to avoid ISI and preserve the subcarriers' orthogonality. On receiver side the guard interval is used for Carrier Frequency Offset (CFO) synchronization. Afterwards it can be removed. In order to transform the data back to the frequency domain using the FFT, again the data stream has to be divided into N parallel data symbols. The equalization in frequency domain is simply a division by the corresponding chan-

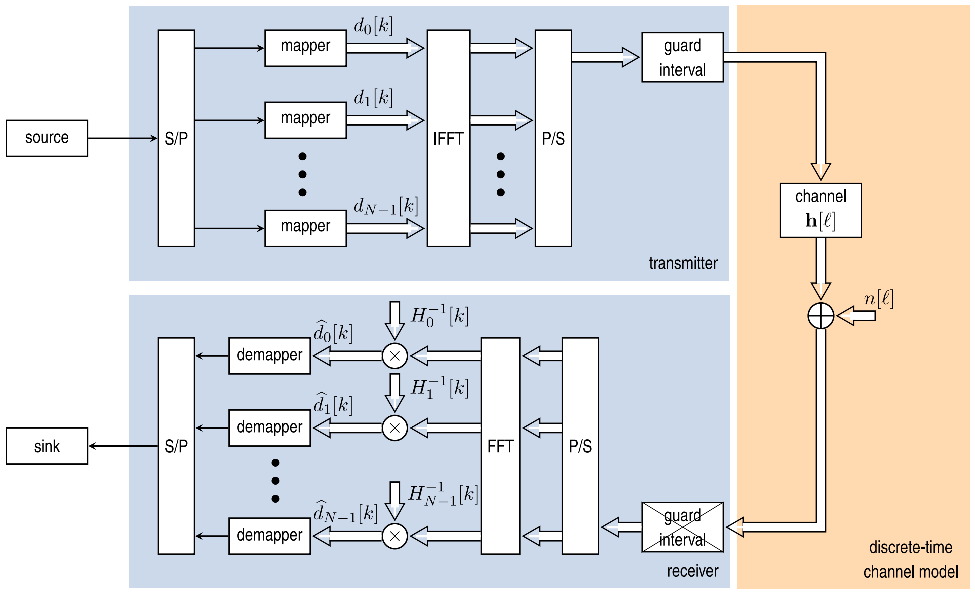

Figure 2: Discrete-time OFDM system model

nel coefficients, since each subcarrier is assumed to be frequency non-selective. Each symbol is then demapped, together forming the original bit stream.

Guard Interval As shown in Figure 2 the guard interval is added after transforming the data from frequency to time domain. There are two reasons why the guard interval is essential for OFDM. As depicted in Figure 3 successive OFDM symbols in time domain do not have a Dirac like impulse function. Therefore, to avoid ISI the next OFDM symbol cannot be transmitted until the impulse function of the previous symbol is zero. This determines the length of the guard interval. The content of the guard interval has to be a cyclic prefix of the current OFDM symbol. The reason for this is that the DFT on receiver side assumes a periodic signal in time domain. Therefore, a cyclic convolution instead of a linear convolution is required, which can be achieved by inserting cyclic prefixes.

Optional questions for self control

- What is a frequency selective channel?
- What effect do we observe when transmitting over frequency selective channels?
- Which approaches do we get to know during the lecture to deal with frequency selective channels?
- What is the optimal detector? How does it work? Why do we need other techniques?
- What is the basic idea behind OFDM? How does the discrete-time OFDM system look like? Explain every block!
- Why do we need a guard interval? How does it look like?
- How does the equalization in OFDM work?
- What drawbacks does OFDM have?

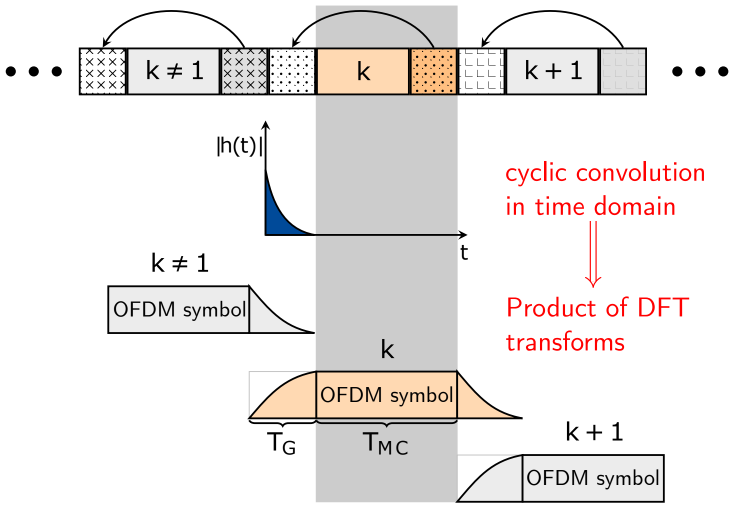

Figure 3: Guard interval of an OFDM symbol

2 Introduction to the Lyrtech Hardware

2.1 Hardware - Physical Setup

As mentioned before, the project is based on a hardware-in-the-loop concept. While the physical transmission is performed using a dedicated communication hardware, transmitter and receiver algorithms are implemented in software running on a host PC.

The provided hardware (HW) consists of a host PC and the following three boards which contain Field Programmable Gate Arrays (FPGAs), Digital Signal Processors (DSPs) as well as Analog-Digital (AD) and Digital-Analog (DA) converters, respectively.

- VHS-AD-Board (AD converter with FPGA),
- VHS-DA-Board (DA converter with FPGA),
- SMQ-Board (2 FPGAs with two DSPs).

All boards are connected via a rapid channel. The DSPs are connected via fastbus with the respective FPGAs. FPGAs will be programmed with synthesized *.bit files, DSPs with compiled by C code *.out files. These files are provided in your working directory so that you can work solely in MATLAB. An appropriate MATLAB script to initialize the HW is provided too.

Furthermore, AD/DA converters are connected with a Radio Frequency (RF) front end to transmit signals in the 2.4GHz band and 5GHz, respectively. In this project, we will use the 2.4GHz band which provides 13 channels with 20MHz bandwith each. There are four transceivers available. The one connected to channels 1 and 2 of the respective DA or AD converter are used for transmission. To measure the signals in base band (Inphase (I) and Quadrature (Q) component), oscilloscopes are used between the converters and the RF front end.

Parameter	Value	
Channel bandwidth	$B = 20\text{MHz}$	(obtained for 64 subcarriers)
Signal bandwidth	$B > 16.25\text{MHz}$	(52 used subcarriers)
Symbol duration	$T_{MC} = 3.2\mu\text{s}$	$\rightarrow$ spectral loss: 25%
Subcarrier spacing	$\Delta f = T_{MC}^{-1} = 312.5\text{kHz}$	
Guard interval	$T_G = 0.8\mu\text{s}$	(16 subcarriers for guard interval)

Table 1: OFDM Parameters of IEEE 802.11a Standard

2.2 MATLAB as Interface to the Hardware

In this project, functions and libraries to program the HW for the transmitter and the receiver are provided. Initializing and programming of the HW is done implicitly in "trans_OFDM_frame" which is the interface to the HW for you. It calls a function to initialize the HW and sends a previously in "create_OFDM_frame" generated transmit frame. As MATLAB per default works with values of type *double*, the frame has to be transformed with "siso_trans_complex_to_byte". This function stores the imaginary part in the upper and the real part in the lower 16 bit of a *uint32* data word because components I and Q are processed in parallel on the HW. Finally the interface to the HW is closed.

The frame which you write into the HW must occupy a fixed length of 8000 values. This frame will be interpreted by the HW as a with 20.8Msps sampled signal. This is tremendously important for generating signals in MATLAB. Actually, the HW works with a sample rate of 104MHz but the signal will be oversampled by a factor of 5 (configurable) on one of the FPGAs. To measure the signals in base band, an oscilloscope is connected to the outputs of the DA converter on channels 1 and 2.

3 Multicarrier Digital Communication System - OFDM

3.1 OFDM Parameter

The main task of this project is to implement an OFDM system based on the Wireless Local Area Network (WLAN) standard 802.11a. The transmit filtering is implemented on the HW, hence is not addressed in this project. Before you start with the implementation, you need to be familiar with the parameters of the standard. These parameters are essential to start the implementation in MATLAB. An overview is given in Table 1.

The IEEE standard 802.11a signal consists of 64 subcarriers. However, just 52 subcarriers are actively used for the signal generation. The other subcarriers are unused to fulfill spectral mask constraints. The spectral mask is necessary to reduce the interference to neighboring bands caused e.g. by out-of-band radiation. Within the 52 used subcarriers, the IEEE standard 802.11a provides 4 subcarriers to transmit pilot symbols. These pilots are not used for channel estimation but for fine synchronization of the residual CFO in the frequency domain. In summary, we have 48 subcarriers for data transmission in one OFDM symbol. Figure 4 depicts all 64 subcarriers and their usage of an OFDM symbol. Note that the subcarrier on position zero represents the DC component. It remains unused to allow the use of simple (cheaper) direct-conversion RF receivers.

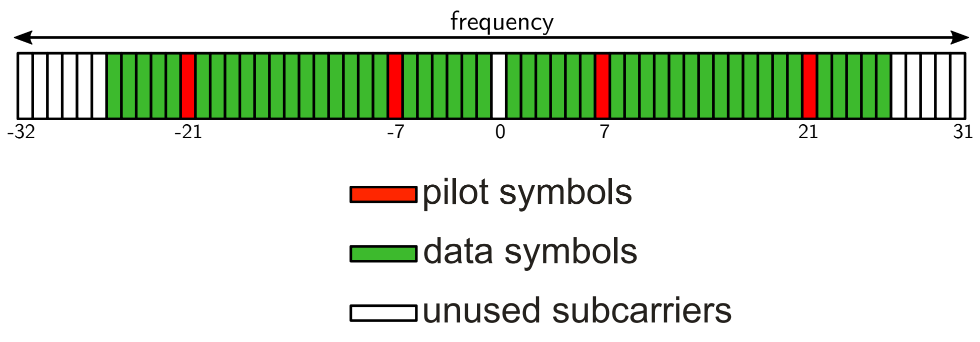

Figure 4: OFDM Symbol

3.2 OFDM Transmitter

During this project we want to transmit a picture of *Lenna* (96x96 pixels, 8 bit gray levels $\rightarrow$ 73728 bit) using OFDM and the IEEE standard 802.11a. Following the previous subsection, the data of this picture has to be transmitted within the 48 subcarriers used for data transmission. However, this picture is too large to be transmitted in a single OFDM symbol. Moreover, for real transmission on the HW some extra processing steps to enable time synchronization and channel estimation at the receiver is needed. Therefore, a specific frame structure of OFDM-Symbols containing Short Term Preamble (STP), Long Term Preamble (LTP), training as well as data symbols is given as follows:

$$x_{\text{Frame}} = [2 \times \text{STP}, 1 \times \text{LTP}, 1 \times \text{TRAINING}, 130 \times \text{DATA}] \tag{1}$$

The STP is used at the receiver HW to just detect the presence of a useful signal. The LTP is used to do the time synchronization, which is already done on the HW. STP and LTP will be cut off inside the HW after receiving a frame. The training data is used for channel estimation. Each frame contains 130 data symbols. This number is defined by the HW. The STP, LTP and training data are provided in the script "create_OFDM_frame". Figure 5 depicts the complete frame structure in frequency domain. Note that the order of the subcarriers is changed compared to Figure 4. The reason for this is that the output of the FFT in MATLAB is shifted in order to start with the zero frequency. Therefore, the negative frequencies are appended in decreasing order at the end of the increasing positive frequencies. Hence, the highest positive and negative frequencies are located in the centered subcarriers.
The pilot symbols are used for fine synchronization of the residual CFO in the frequency domain. They are located in this project on carriers $(8, 22, 44, 58)$ and should be filled with appropriate symbols, e.g., fixed Quarternary Phase Shift Keying (QPSK) symbols which you can copy from the training sequence.

To start the MATLAB implementation make sure your working directory is "OFDM_Transmitter" which is a sub-directory of 'DC_SS2023'. The first thing to do is to create the OFDM frames containing the *Lenna* picture located in your working directory. Use the the script "create_OFDM_frame". The first part of this script is already implemented. It adds path variables to the sub-directories located in you working directory and defines different OFDM parameter. The "framework" directory contains different functions that might be useful:

- generate_mapping $\rightarrow$ used to create a mapping scheme
- map_symbol $\rightarrow$ maps binary data to symbols
- demap_symbol $\rightarrow$ demaps symbols to binary data
- de2bi_ $\rightarrow$ transforms decimal numbers to binary
- bi2de_ $\rightarrow$ transforms binary numbers to decimal

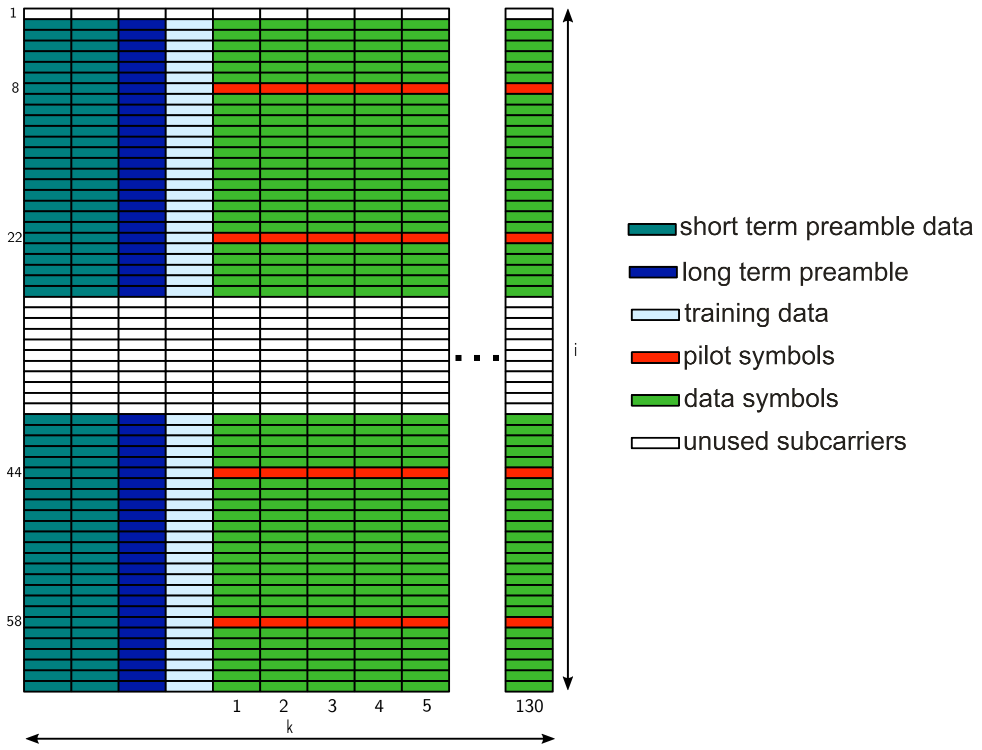

Figure 5: OFDM Frame

- get_preamble_and_training → defines preamble and training data
- predefine_parameter → defines OFDM parameter based on IEEE standard 802.11a

Your task is divided into 6 small parts. For each part or Working Package (WP) a MATLAB function has to be implemented, to provide the necessary functionality. These functions are predefined in the directory "functions_for_WPs", specifying the input and output parameter as well as a short documentation. If you have troubles, you can also use provided pcode files, which are content-obscured, executable files. They implement the same functionality as required in WP.

Task 3.1

WP1: Import the *Lenna* picture and transform the data to QAM symbols. Use the variable "mapping", which defines a QAM mapping, and the framework function "map_symbol.
Hint: The following MATLAB functions could be useful: imread(), bin2dec() or bin2dec_() (framework), map_symbol() (framework).

WP2: Rearrange the data to match the structure of OFDM symbols. Use the predefined variables in the struct "parameter".
Hint: The variable "Nused_all_idx" defines the used carriers, "Nused_data_idx" the position of data and "Nused_pilot_idx" the position of the pilot symbols within one OFDM symbol.

WP3: Add pilots to your OFDM symbols in the correct position defined in "Nused_pilot_idx". Use training data for this.

WP4: Create the frame structure depicted in Figure 5. Use the predefined STP, LTP and training data. Since we need more than one frame to transmit the *Lenna* picture, we need additional information to find out the correct order of frames at the receiver. Therefore, we introduce a header symbol, which will be the first of the 130 data symbols in a frame.

$$x_{\mathrm{DATA}}^{130} = [1 \times \mathrm{HEADER}, 129 \times \mathrm{DATA}] \tag{2}$$

The header symbol will be an increasing number for different frames starting by 1. The output shall be a cell array containing all frames to transmit the picture.

WP5: Transform the frames to time domain.

WP6: Add the guard interval in time domain. The frames are now ready to send.

After creating the frames the actual data transmission can be performed using the script "trans_OFDM_frame". You can ask your supervisor to do the transmission as well as the reception together.

3.3 OFDM Receiver

In the following exercises, you will receive the OFDM frames over one of the HW devices. Therefore, you need to activate your transmitter on another HW device. Make sure your working directory is "OFDM_Receiver" which is a sub-directory of 'DC_SS2023'. For reception, you will use the script "lenna_receiver" which gathers 100 frames received by the HW and stores them in "rec_frame.mat". You can find the script in your current working directory.

After receiving and storing the frames in "rec_frame.mat" you have to process this data to reconstruct the actual picture. This is done in the script "process_recv_frame". Again the first part of this script is already implemented. Your task is the reconstruction of the received picture in 11 small steps. Similar to the transmitter, each step contains a WP in which you have to implement a predefined function already stored in the directory "functions_for_WPs".

Task 3.2

WP7: Rearrange the data of a specific frame in order to match the familiar frame structure. Note that the HW already cut the STP and LTP.

WP8: Estimate the Fractional Frequency Offset (Fractional Frequency Offset (FFO)) part of the CFO. This is done by the so called "Schmidl-Cox algorithm". It exploits the OFDM structure in time domain, i.a. the guard interval contains the cyclic prefix. The phase shift due to FFO is determined by:

$$\hat{\epsilon}_{\text{FFO}} = -\frac{1}{2\pi}\arg\left\{\sum_k \sum_{\mu=-N_G}^{-1} y_\mu[k]\cdot y^*_{\mu+N}[k]\right\} \tag{3}$$

where k is a specific OFDM symbol, N_G is the length of the guard interval and N is the length of the core symbol (64 here). Equation 3 correlates the guard interval with the end of the core symbol, which should be equivalent. The negative argument normalized by 2π delivers the average phase shift between samples at distance $T_{MC} = N \cdot T_s$. This is done for every OFDM symbol. The estimates are averaged to improve the FFO estimate.

WP9: Correct the FFO due to the previously estimated offset. Note that the phase of each sample within one OFDM symbol is shifted by $\frac{\hat{\epsilon}_{\text{FFO}}}{N}$. The correction is done by:

$$\tilde{y}[l] = y[l]\cdot e^{-j2\pi\cdot\hat{\epsilon}_{\text{FFO}}\frac{l}{N}} \tag{4}$$

where l is a run index over all samples of all OFDM symbols of one frame.

WP10: Remove the guard interval, since it is not needed anymore.

WP11: Transform the frame back to frequency domain.

WP12: Back in frequency domain we can do the channel estimation by means of training symbols. The channel estimation is done as follows:

$$\hat{H}_i = \frac{\tilde{Y}_i^{\text{training}}}{d_i^{\text{training}}} \tag{5}$$

WP13: Now you have to equalize the received OFDM symbols in order to remove the effect of a frequency selective channel. The advantage of OFDM is that the equalization can be performed in frequency domain. Since the channel of each subcarrier is supposed to be not frequency selective anymore, the equalization is just a simple division by the corresponding channel coefficient:

$$\hat{d}_i[k] = \frac{\tilde{Y}_i[k]}{H_i} \tag{6}$$

WP14: Use the pilot symbols located on carriers $(8, 22, 44, 58)$ to estimate and remove the Residual Frequency Offset (RFO) part of the CFO. The RFO is not dependent of the specific subcarrier. Hence, using 4 carriers for pilot symbols just improves the reliability of the RFO estimate. First, determine the RFO for each pilot carrier ν:

$$\hat{a}^{\nu}_{\text{RFO}}[k] = \frac{\hat{d}_{\nu}[k]}{d_{\nu}} \tag{7}$$

Note that d_{ν} denotes the predefined symbol sent on the pilot carrier. In this case we used training symbols. The correction can be performed in different strategies, specified in the parameter "mode":

average: The value to shift each sample of one OFDM symbol is determined by averaging the RFO over all pilots:

$$\hat{\epsilon}_{\text{RFO}}[k] = \frac{1}{2\pi}\text{arg}\left\{\frac{1}{4}\sum_{\nu}\hat{a}^{\nu}_{\text{RFO}}[k]\right\} \tag{8}$$

weighted average:The value to shift each sample of one OFDM symbol is determined by weighted averaging the RFO over all pilots. The weights are defined by the corresponding channel coefficients:

$$\hat{\epsilon}_{\text{RFO}}[k] = \frac{1}{2\pi}\text{arg}\left\{\frac{\sum_{\nu}|H_{\nu}[k]|^2\cdot\hat{a}^{\nu}_{\text{RFO}}[k]}{\sum_{\nu}|H_{\nu}[k]|^2}\right\} \tag{9}$$

none: No correction is done.

For each mode (except *none*) the correction can be performed as follows:

$$\tilde{d}_i[k] = \hat{d}_i[k]\cdot e^{-j2\pi\hat{\epsilon}_{\text{RFO}}[k]} \tag{10}$$

After correcting the RFO you can remove all carriers which are not used for data transmission.

WP15: Classify each frame, i.e. detect the header symbol and store it in the correct order. **Hint:** The following MATLAB functions could be useful: bin2dec() or bin2dec_() (framework), demap_symbol() (framework).

WP16: Concatenate each frame to create a common data structure. Demap each QAM symbol to binary representation. **Hint:** The following MATLAB functions could be useful: demap_symbol() (framework).

WP17: Reconstruct the gray levels using the binary data representation. Use uint8 for decimal values, i.e. 8 bit for one decimal number with 'left-msb'. Reconstruct the original picture with 96x96 8 bit gray levels.

A Abbreviations

AD	Analog-Digital
QPSK	Quarternary Phase Shift Keying
CFO	Carrier Frequency Offset
FFO	Fractional Frequency Offset
RFO	Residual Frequency Offset
DA	Digital-Analog
DSP	Digital Signal Processor
FPGA	Field Programmable Gate Array
HW	hardware
I	Inphase
LTP	Long Term Preamble
OFDM	Orthogonal Frequency Division Multiplexing
RF	Radio Frequency
Q	Quadrature
STP	Short Term Preamble
WLAN	Wireless Local Area Network
WP	Working Package