1 ICT Laboratory Overview - CIT Master

1.1 Introduction

The ANT part of the ICT laboratory held in the winter term is meant to be solved in groups of two in an independent fashion with minimal help from tutors. You are expected to solve problems on your own and organize your work as you see fit. To provide time for questions, initial instructions and to evaluate your implementation 4 laboratory dates of approximately 4 hours each and a short setup meeting are scheduled.

CIT masters are expected to implement a baseline receiver that matches the system described in this lab description. The transmitter will be provided in form of Matlab p-code to test and simulate the whole transmission chain. Additionally, a short presentation about one of the transmitter/receiver blocks has to be presented.

In the following, Section 1.2 discusses the specific goals and requirements of this lab in more detail. Then, Section 1.3 introduces the lab dates and the general timing of the lab over the whole winter term. The explanations of the specific tasks for phase 1 and 2 are given in Section 2. Finally, Section 3 explains the evaluation guidelines that will be used to judge if the lab has been passed successfully or not.

1.2 Goals and Requirements

1.2.1 Requirements

This laboratory is mandatory for CIT master students. Besides different Bachelor backgrounds we expect you to have certain knowledge and skills at the beginning of the laboratory. To some degree, it is expected that you will have to research topics less well known to you, but nonetheless the following is expected:

- Self-motivated working style (researching unknown topics with minimal tutor help)
- Basic communications technology knowledge (equivalent GNT from the German Bachelor ET/IT)
- Basic knowledge of MATLAB
- Basic knowledge of presentation techniques / software (e.g., LaTeX Beamer or Powerpoint)

1.2.2 Goals

The following goals are targeted with this laboratory:

- Self-motivated problem solution including research and collaboration with other lab attendees
- Deepening knowledge about all the basic processing steps in communications from digitization to coding and modulation
- Developing programming skills in Matlab
- Practicing presentation of technical details / procedures (written and oral)

1.3 Time line

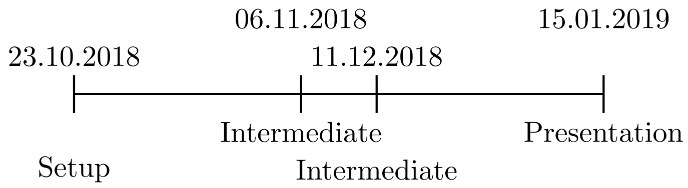


Figure 1: Time line for ICT Lab 1 WS2017/2018.

This laboratory is planned to be running alongside other courses during the whole winter term with only 5 predetermined lab dates for an overall workload of 1CP or 30h for CIT masters. Figure 1 shows the specific dates for WS 2017/2018 and their individual purpose. The first date is mandatory for all students attending the lab and will be used to clarify the setup, goals and requirements again. It is expected that all students read this script beforehand and are well prepared. After this initial meeting, intermediate lab time is offered to continue the work on the task, ask questions and get help with individual problems. All intermediate dates are offered as supervised implementation time to help you with problems and give pointers towards helpful material. For CIT masters attending these dates is not mandatory. However, we strongly advise you to use this time to solve problems and get feedback!

The final date of the lab is mandatory and will be used to check the fulfillment of the tasks and to present the findings of each group in a 5-7 minute presentation of maximum 5 slides. The tutors will test each groups implementation according to the API requirements described in Section 2.3.2, check the code for proper comments and fulfillment of the test defined in Section 3.4. **Groups who do not pass this test will have to repeat ICT lab 1 in the following year**.

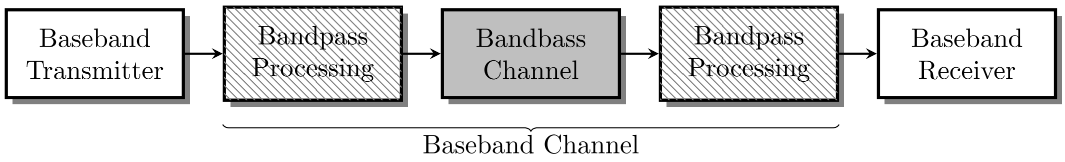


Figure 2: Overview of a point-to-point communication setup. Shaded and gray marked blocks will be provided.

2 Task Description

2.1 General Description

The general idea of this lab is the implementation of a complete point-to-point communication chain as illustrated in Fig. 2, including transmitter, channel and receiver. To restrict the breadth of this task, CIT masters only have to implement the receiver parts indicated by white blocks. An equivalent baseband channel model and the transmitter will be provided to test the overall communication chain. This model summarizes all channel and hardware effects that are attributed to bandpass processing, including but not limited to up/down conversion, amplification, antenna patterns, and so on. However, some of the bandpass effects will be included into the lab by equivalent baseband descriptions as “non-linear hardware” (see the following sections for more details).

Note: Read the transmitter section carefully! You need all information in both transmitter and receiver sections to implement the communication system successfully.

2.2 Transmitter Structure for Reference

2.2.1 Transmitter Model

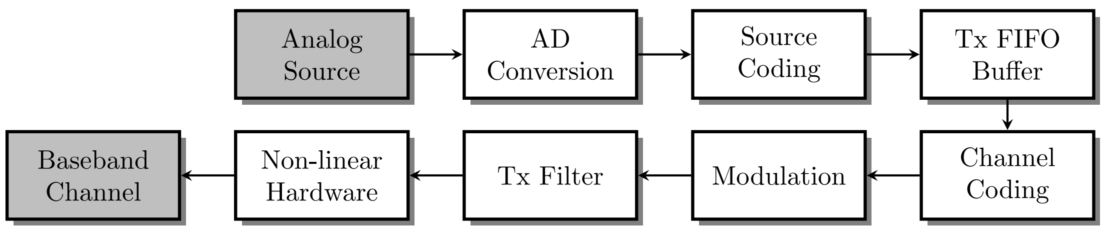


Figure 3: General structure of the Baseband Transmitter as introduced in Fig. 2 with interface to Baseband Channel. The transmitter blocks will be provided as p-code.

The transmitter of a basic point-to-point communication chain according the the specifications below is available as p-code. The specifications here are for reference and to ease implementation of the complete chain as well as the receiver. Fig. 3 shows the building blocks of such a transmitter. Each block is defined by its inputs and outputs and a short requirements list that describes the functionality in Section 2.2.2. Please note, that some blocks are marked as “switchable” by a parameter `switch_off`, which means that such a block should not change the input data in any way if switched “off” by `switch_off=1`, i.e., output=input.

Additionally to functional requirements, e.g., a certain quantization of the data in the AD conversion block, also optional graphical output may be available. For example, the quantization error introduced in the AD Conversion may be plotted in a figure. Graphical output should always be optional, i.e., controlled by a switching variable `switch_graph`, to analyze your implementation and the results as needed.

2.2.2 API Definitions

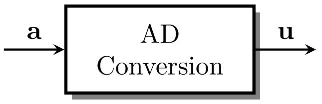
```
a=analog_source(par_no,switch_reset,switch_graph);
```

Pseudo-analog source providing a highly oversampled signal for further processing. Repeated calls of this function will provide consecutive blocks of the signal.

Parameters:

`par_no`	indicates the number of (oversampled) samples to provide.
`switch_reset`	if equal to 1, the source is reset to the beginning.

```
u=ad_conversion(a,par_w,par_q,switch_graph);
```

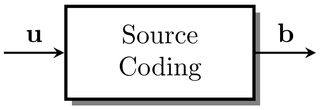


This block facilitates AD Conversion, i.e., sampling and quantization.

Requirements:

1. Downsampling by a factor `par_w` according to the bandwidth of the input signal **a**.
2. Quantization of the signal's amplitude to `par_q=8` bits.
3. This function should return the bit representation as row vector for each sample.

```
[b,code_tree]=source_coding(u,par_scblklen,switch_off,switch_graph);
```



This block facilitates Source Coding according to Huffman.

Requirements:

1. This function expects the quantized and sampled signal in bit representation as row vectors per sample.
2. Analyze the quantized and sampled signal of length `par_scblklen` to build the code tree `code_tree`.
3. Encode the block of length `par_scblklen` of the quantized signal **u** into a binary representation **b**.
4. Due to the variable compression rate, the length of the output sequence `b` will not be constant and needs to be stored.

Assume a block length of `par_scblklen=100`.

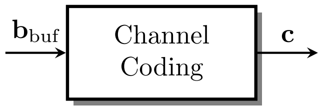
`[b_buf]=tx_fifo(b,par_fifolen,par_ccblklen,switch_reset);`

This block facilitates First-In First-Out (FIFO) buffering of source coded bits to ensure correct block lengths for further processing.

Requirements:

1. Internally store bit vectors `b` in a buffer. If buffer fill equals or exceeds `par_ccblklen`, return `par_ccblklen` bits as vector `b_buf` and remove these from the internal buffer. If buffer fill is less, return an empty vector.
2. Avoid buffer overruns, i.e., make sure that `par_ccblklen` is larger than length of `b`.
3. If `switch_reset=1`, empty buffer before filling.

`c=channel_coding(b_buf,par_H,switch_off);`



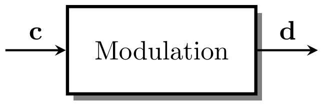
This block facilitates Channel Coding by a [7,4] Hamming code.

Requirements:

1. Encode the block of length `par_ccblklen` of the binary signal $\mathbf{b}_{\text{buf}}$ via the [7,4] Hamming block code.

The block length should be a multiple of 16 bits.

`d=modulation(c,switch_mod,switch_graph);`



This block facilitates Modulation of the encoded bit sequence to either 16-QAM or 16-PSK.

Requirements:

1. Modulate data to either 16-QAM or 16-PSK with Gray mapping. `switch_mod=0` indicates 16-QAM, `switch_mod=1` 16-PSK.
2. Normalize the average symbol power to 1.

The block length should be a multiple of 7 symbols.

`s=tx_filter(d,par_tx_w,switch_graph);`

This block facilitates filtering of the digital symbols with a digital ideal low-pass filter.

Requirements:

1. Filter a block of symbols with an ideal low-pass filter using an oversampling factor of `par_tx_w=8`.
2. Normalize the filter output signal appropriately to ensure distortion free operation considering the non-linear hardware.

`x=tx_hardware(s,par_txthresh,switch_graph);`

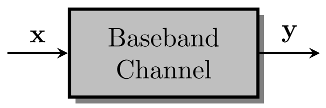
This block models the influence of an amplifier on the baseband signal by hard thresholding.

Requirements:

1. Implement a simple hard thresholding function that limits the *absolute value* of the baseband signal `s` such that it is linearly scaled to be smaller than 1 up to values of `par_txthresh` and clipped to 1 if greater than `par_txthresh`.
2. Ensure that the phase of `s` is not changed by this block.
3. Analyze distortions by this block if the scaling after Tx Filtering is suboptimal and switch from 16-QAM to 16-PSK.

Assume a standard parameter `par_txthresh=1`.

`y=channel(x,par_SNRdB,switch_graph);`



This block models a simple baseband channel that adds white gaussian noise to the signal.

Parameters:

`par_SNRdB` will be used to check the performance of the transceiver chain at different SNRs (in dB).

2.3 Receiver implementation

2.3.1 Receiver Model

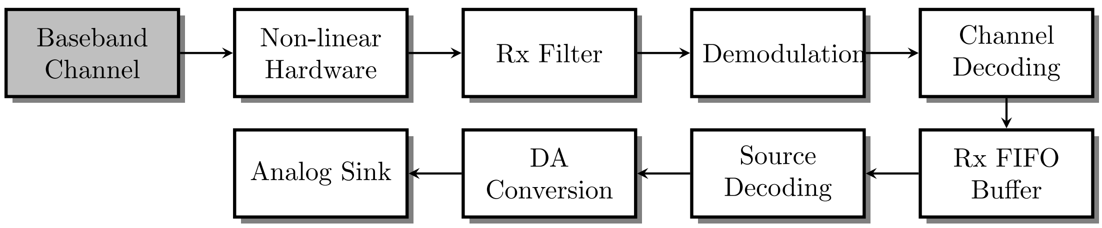


Figure 4: General structure of the Baseband Receiver with interface to Baseband Channel. Gray blocks will be provided, white ones are to be implemented according to the specifications.

The task of this lab comprises the implementation of the baseline receiver for an AWGN channel and the overall simulation. Fig. 4 shows the building blocks of such a receiver and Section 2.3.2 details the individual blocks in terms of inputs, outputs and requirements. To simplify the task some parameters can be assumed as known at the receiver side, i.e., the `code_tree` for each Huffman encoded block is known and the scaling of the transmit signal is also known. This also applies to modulation, channel code and output block length of source coder.

Your task is the fulfillment of these requirements for each block while adhering to the specified inputs, outputs and function names. Please note, that some blocks are marked as "switchable" by a parameter `switch_off`, which means that such a block should not change the input data in any way if switched "off" by `switch_off=1`, i.e., output=input.

In addition to the receiver implementation a simulation environment has to be created that uses the transmitter and receiver implementations to numerically analyze the performance of the whole point-to-point communication chain. The following requirements have to be fulfilled:

- Allow simulation of different SNRs, e.g., using an outer loop.
- Save the results in terms of uncoded/coded bit error rate (BER) and mean square error (MSE) for different SNR choices in a vector.
- Plot the BER and MSE vs. the SNR.

2.3.2 API Definitions

`s_tilde=rx_hardware(y,par_rxthresh,switch_graph);`

This block models the influence of an amplifier on the baseband signal by hard thresholding.

Requirements:

1. Implement a simple hard thresholding function block that is transparent and does not change the signal in any way.

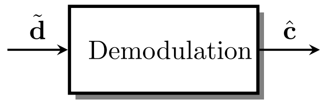
`d_tilde=rx_filter(s_tilde,par_rx_w,switch_graph);`

This block facilitates filtering of the received signal with a digital ideal low-pass filter.

Requirements:

1. Filter the received signal with an ideal low-pass filter using an downsampling factor of `par_rx_w=8`, i.e., identical to the transmitter side.
2. Normalize the filter output signal appropriately to ensure that the power of the signal is not changed.

`c_hat=demodulation(d_tilde,switch_mod,switch_graph);`

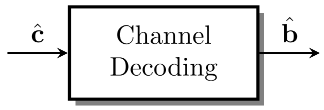


This block facilitates hard estimation of the code bits for either 16-QAM or 16-PSK.

Requirements:

1. Decide the received signal to either 16-QAM and 16-PSK symbols with Gray mapping to estimate the code bits.
2. Ensure proper processing in terms of the channel and source encoded blocks afterwards.

`b_hat=channel_decoding(c_hat,par_H,switch_off);`



This block facilitates Channel Decoding of a [7,4] Hamming code.

Requirements:

1. Correct errors in the estimated code words of the [7,4] Hamming block code by syndrome decoding.

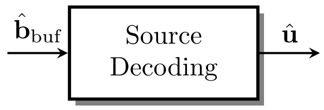
`b_hat_buf=rx_fifo(b_hat,par_fifolen,par_sdblklen,switch_reset);`

This block facilitates buffering and segmentation of decoded bits.

Requirements:

1. Collect bit vectors `b_hat` in an internal buffer and return them segmented into the correct block lengths for subsequent source decoding.
2. The correct block lengths of the output `b_hat_buf` are determined by the stored source coded message `b` from the `source_coding` step in the transmitter, which means the correct block length `par_sdblklen = length(b)`.

`u_hat=source_decoding(b_hat_buf,code_tree,switch_off);`

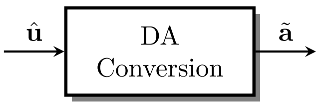


This block facilitates Source Decoding of the Huffman encoded and hard estimated source bits.

Requirements:

1. Decode each block of length `par_sdblklen` bits of the quantized signal using the `code_tree` of the block.
2. Return the bit representation of the decoded signal.

`a_tilde=da_conversion(u_hat,par_w,par_q,switch_graph);`



This block facilitates DA Conversion, i.e., upsampling and reconstruction.

Requirements:

1. Upsampling by the factor `par_w` that has been chosen at the transmitter side.
2. Reconstruct the Pseudo-analog source signal.

`[MSE,BER]=analog_sink(a,a_tilde,b,b_hat,...);`

Processing of the reconstructed and original signal to analyze the errors due to transmission. Here, the analog sink represents the analysis of the received and reconstructed signals. Knowledge of all other signals in the system is implicitly assumed.

Requirements:

1. Calculate the error in terms of the mean square error (MSE)
2. Calculate the error in terms of the coded and uncoded bit error rate (BER).

3 Evaluation Guidelines

3.1 General Rules

Besides the solution of the task that is detailed below, we expect you to adhere to some general rules:

- Solve the tasks by yourselves.
- Write your own code and do not copy!
- Design your own slides and do not copy (pictures, too)!

Group efforts in solving the sub tasks are encouraged and expected, but we will collect the solutions of all groups at the end of the lab and test your personal knowledge about your solution. The goal of this lab is to enhance your ability to break down bigger tasks into smaller steps, organize your work and research for yourself. If you just copy the solution of other groups, you will simply limit your own benefit.

3.2 What to expect from the tutors?

The tutors will help you understand the tasks, may give you help finding the right information and evaluate your work to judge if you have passed or not.

Most importantly:

- Tutors will ***not*** write Matlab code for you!
- Tutors will give you hints and tips to help you to *find the solution yourself*!
- Tutors will only help you if you *follow the guidelines* and API descriptions given in this document!

3.3 Required Performance

To pass ICT lab 1 the following expectations have to be met. Except the compliance test, which is a hard measure checked in Matlab, compliance is rated by the tutor:

- Compliance with the tests described in Section 3.4 is mandatory to pass the lab.
- We expect you to write clean **and** well documented Matlab code that is easily readable by the tutor. Consider this lab to be part of a job that will be carried on by another team after you finish.
- Additionally to the compliance test, the tutor may ask you questions about your implementation to test your individual grasp of the solution.
- A short presentation about parts of the baseline transmitter/receiver in front of all other groups of maximum 5 slides taking 5-7 minutes is expected. Therein, you should quickly explain: (1) the problem, (2) your approach, (3) the solution and (4) the final results.

3.4 Receiver Compliance Test

The compliance test for the receiver comprises the following checks:

Non-linear Hardware
1. Correct clipping characteristic
2. Figure of received signal and signal after hardware showing that no clipping is in effect

Rx Filter
1. Figure of the filter output
2. Figure showing eye pattern

Demodulation
1. Correct demodulation with Grey mapping
2. Figure of the estimated symbols with decision thresholds

Hamming Decoding
1. Correct channel decoding by syndrome decoding

	2. Figure of exemplary code word indicating corrected errors
Rx FIFO	1. Correct reconstruction of source encoded frames
Huffman Encoding	1. Correct Huffman decoding
DA Conversion	1. Correct upsampling 2. Figure indicating the reconstructed signal