DESIGN SPECIFICATION

Simon Sörman

Version 1.0

Status

Reviewed	Antonios Pitarokoilis	09/10/2015
Approved	Antonios Pitarokoilis	14/10/2015

PROJECT IDENTITY

HT 2015
Linköping University, ISY

Participants of the group

Name	Responsible	Phone	E-mail
Miguel Abadia	Software Engineer	073 723 98 37	migab416@student.liu.se
Herman Molinder	Project Manager	076 823 81 96	hermo276@student.liu.se
Simon Pålstam	Software Engineer	076 803 17 06	simpa265@student.liu.se
Thiti Sookyoi	Software Engineer	073 573 66 63	thiso311@student.liu.se
Simon Sörman	Responsible for the documentation	070 954 78 41	simso657@student.liu.se

Customer: Mikael Olofsson, mikael.olofsson@liu.se, 013 – 28 13 43
Examiner: Danyo Danev, danyo@isy.liu.se, 013 – 28 13 35
Supervisor: Antonios Pitarokoilis, antonios.pitarokoilis@isy.liu.se, 013 – 28 13 40

Contents

Document history

Version	Date	Changes	Performed by	Reviewed
0.1	06/10/2015	First draft	All group members	MA,SS,HM
0.2	13/10/2015	• Corrected grammar, typos and wordings • Changed mathematics • Changed input and output of functions	All group members	SP,HM
1.0	14/10/2015	• Corrected wordings • Changed name on block “Signal modulator” to upconverter	HM,SS	HM

1 INTRODUCTION

In this project, which is part of the course TSKS05 – Communication Systems CDIO, the students are to construct a system which demonstrates parallel data transmission of massive MIMO, the ability to transmit signals intended towards different users over the same time and frequency resources. Massive MIMO is a hot topic in wireless communications and a candidate for the fifth generation cellular systems (5G).

This project is a continuation of a former project in the same course. The former project group built the hardware of an array of loudspeakers and microphones and developed software for transmission and reception of audio signals. This project aims to further develop this system into transmitting digital data to two terminals in the same time and frequency by the use of the existing hardware.

Note that all implemented functions and theoretical justification will account for a general number of terminals to enable an easy expansion in the future. But the complete system will only be developed and tested for two terminals.

The purpose of this document is to provide a detailed description of the system design.

1.1 Definition of terms

In table 1 below, abbreviations used throughout the document is listed and explained.

Table 1: Abbreviations used in the document.

Word	**Definition**
A/D	Analog to Digital
BER	Bit Error Rate
BPSK	Binary Phase Shift Keying
D/A	Digital to Analog
DF	Decision Feedback
GUI	Graphical User Interface
ISI	Inter-Symbol Interference
L/M	Loudspeaker/Microphone
LSE	Least-Square Estimation
MIMO	Multiple Input Multiple Output
MLE	Maximum Likelihood Estimation
MMSE	Minimum Mean Square Error
MRT	Maximum-Ratio Transmission
QPSK	Quadratic Phase Shift Keying
TDD	Time Division Duplex

ZF	Zero-Forcing
MIMO-array	The array of L/M units that are used for beamforming
MIMO-transceiver	One L/M unit in the MIMO-array

2 OVERVIEW OF THE SYSTEM

The complete system consists of both hardware and software implementations. This project only focuses on the software part of the complete system, as the hardware that will be used was created during the last year's project. The hardware consists of a computer, 8 L/M pairs, A/D and D/A converters and a distribution box, as described in last year's project documentation (Stenmark, 2014a). The software will be divided into three sub-systems, the channel estimator, the modulator and the channel coder. A simple overview of the software sub-systems is shown in Figure 1.

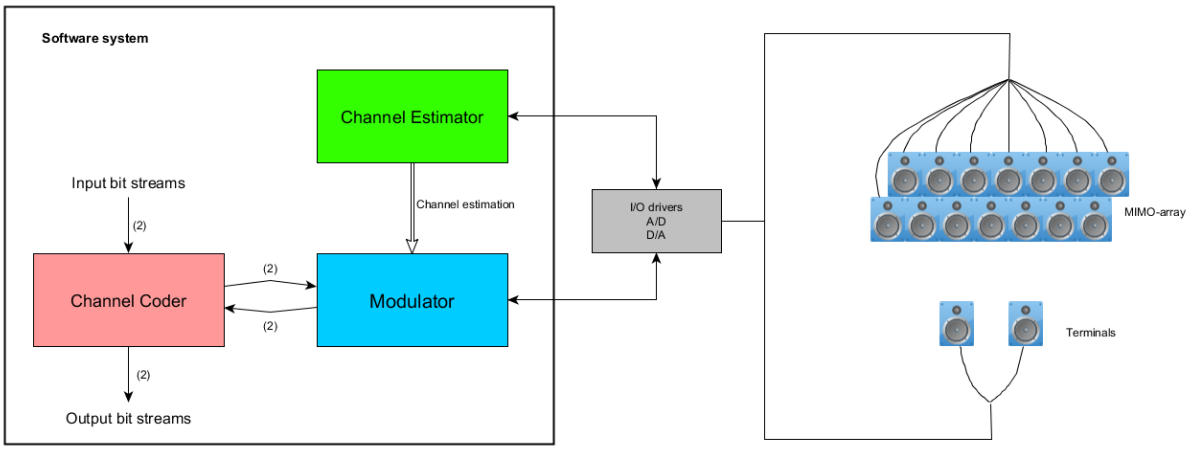

Figure 1: Overview of the sub-systems.

The interaction of the various processing blocks for the communication is shown in Figure 2:

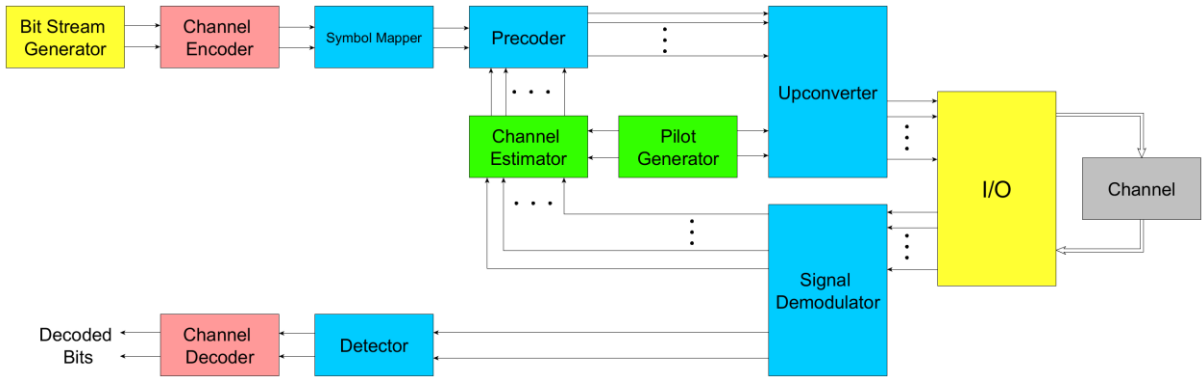

Figure 2: Overview of the software system.

In the figures, the blocks are color coded. Red blocks are part of the channel coder, the blue ones are the modulator and the green blocks are the channel estimator. Yellow blocks are not part of a sub-system.

As shown in Figure 2 the software system will function as follows. The channel encoder will be presented with two bit streams that in the end are to be transmitted from the MIMO array to the two terminals. The coder will encode these bit streams and output two coded bit streams. The channel estimator will estimate the channels. In order to do this each terminal sends one pilot signal to the array. The received signals during pilot transmission are passed on to the channel estimator and processed to get channel estimates. These estimates will be used by the precoder, for appropriate prefiltering of the (coded) information streams. These signals will be modulated by the upconverter to be the real valued signals that are transmitted. The signal demodulator does the reverse operation; it takes the real sampled signals and produces the complex baseband representation. The detector converts the baseband signals to fit the channel decoder. The channel decoder will use this information to estimate the most likely information bit sequences.

The bit stream generator's only task is to provide streams of bits that the system should use. This could be random bits or representations of text messages that the user could specify, or something else. The I/O-block is responsible for controlling the hardware, making sure that the upconverter/demodulator has an easy to use interface to the L/M-pairs.

2.1 Main Program

The main program will be a script with a text based user interface in the Matlab terminal. The program will start by running the script and then choosing a few parameters. When the test is done the result will be presented in the Matlab Terminal and in several graphs.

The basic structure of the main program is that, given some initialization parameters, it will call the different boxes in Figure 2 in a certain order with some well-defined arguments.

2.2 Included sub-systems

The system will be divided into three sub-systems. The sub-systems will be modular in the sense that the user will be able to change internal function of the sub-systems without affecting the function of the complete system. The system is divided in the following three sub-systems:

1. Channel Estimator
2. Modulator
3. Channel Coder

2.2.1 Other sub-system

Bit stream generator
The bit stream generator will generate 2 bit streams to the system.

Table 1: List of inputs and outputs for the pilot generator.

Input	**Type**
Number of bits	Integer
Number of terminals	Integer
Output	**Type**

Matrix containing bit streams of length (N_b) for each of the *(K)* terminals.	Matrix of $(K\ x\ N_b)$

2.3 Hardware interface

Existing hardware will be used to build the system. The hardware was made by the group who took this course the last year and consists of the following entities:

- A computer with Windows 7 and required drivers installed.
- A/D converter Contec AD12-64 (PCI).
- D/A converter Contec DA12-16 (PCI).
- 8 L/M-pairs
- Distribution box
- Required cables and connectors

It is important to know that the A/D-card has 64 channels and that the D/A-card has 16 channels. Both these cards have a maximum of $f_s = 100$ kSamples/s in total, which has to be split on the used channels. Another limitation of the A/D converter is that one can't specify which channels to use. If one wants to sample on 4 channels, then it will sample on channel 0, 1, 2 and 3. This means that when sampling on a group of L/M-pairs, the sampling frequency is limited by the maximum channel number of that group.

For more information about the hardware, the reader is referred to last year's user manual (Stenmark, 2014a).

2.3.1 External Interface

The Matlab class IOobject will be used for the interface with the hardware. This class will provide the functions listed in table 2:

Table 3: Functions of the IOobject

Function	**Description**
Constructor	Will take the A/D and D/A identifiers and initialize the hardware drivers.
setArrayChannels	Sets a vector *array_channels* of length *M*, which contains the channel numbers used for the MIMO-array.
setTerminalChannels	Sets a vector *terminal_channels* of length *K*, which contains the channel numbers used for the terminals.
setArrayControlGroup	Sets which control group (1 or 2) the MIMO-array is in.
setArraySampleFreq	Sets and returns which sample frequency should be used for sampling of the MIMO-array microphones.
setTerminalSampleFreq	Sets and returns which sample frequency should be used for sampling the terminal microphones.
setArraySendFreq	Sets and returns which sample frequency should be used when transmitting from the

	MIMO-array loudspeakers.
setTerminalSendFreq	Sets and returns which sample frequency should be used when transmitting from the terminal loudspeakers.
sendArrayToTerminal	Takes a matrix of size $M\ x\ N_s$, where row *i* gets sent on channel *array_channels*(*i*). Returns a matrix of size $K\ x\ X_s$where row *j* is received on channel *terminal_channels*(*j*). $X_s$corresponds to the sending time or can optionally be chosen.
sendTerminalToArray	Takes a matrix of size $K\ x\ N_s$, where row *i* gets sent on channel *terminal_channels*(*i*). Returns a matrix of size $M\ x\ N_x$where row *j* is received on channel *array_channels*(*j*). N_s corresponds to the sending time or can optionally be chosen.

The hardware limitations on the A/D and D/A converters gives the following requirements for arguments:

$$(\max(array\_channels)\ + 1) * array\_sample\_freq \leq 100\ 000$$

$$(\max(terminal\_channels)\ + 1) * terminal\_sample\_freq \leq 100\ 000$$

$$M * array\_send\_freq\ \leq 100\ 000$$

$$K * terminal\_send\_freq\ \leq 100\ 000$$

Observe that the hardware that is going to be used always consists of 14 elements in the MIMO-array and two terminals, meaning that $M\ =\ 14$ and $K\ =\ 2$. The software will however support any numbers to enable easy upgrades.

2.3.2 Implementation

At an early stage the IOobject will be implemented using the ML-DAQ library produced by Contec together with the Data Acquisition Toolbox in Matlab. This implementation will therefore be written entirely in Matlab and provide an early interface to the hardware that can be used during testing and the final product if the result is good enough. The early implementation will however suffer some extra limitations that were present in last year's project as well. For example, it won't be able to send signals longer than about 2.5 seconds without a noticeable gap in the transmitted sound, and there is a large risk of the computer to crash, resulting in a blue screen error (BSOD) (Stenmark, 2014b). Another limitation that might appear (depending on how the ML-DAQ library is implemented, which is out of our control) is that we'll always have to use all 16 channels on the converters, which reduce the requirements into:

$$array\_send\_freq, terminal\_send\_freq, array\_sampling\_freq, terminal\_sample\_freq \leq \frac{100000}{16} = 6250$$

If these problems are found to be troublesome for the project or the product, or if time allows, the IOobject functions can be implemented in the C language, using the drivers for the A/D and D/A cards. The C functions will be written so that they can be called from Matlab, and won't change the external interface in any way. Thus this software upgrade can be done independently. By doing this other implementation, we will also get more control over the hardware, which could prove useful.

3 CHANNEL ESTIMATOR

The channel estimator will estimate the channel impulse response between each transceiver in the MIMO array and both terminals. A TDD protocol will be used where the terminals transmit initially pilots and subsequently the transceiver array transmits data to the terminals. Each transceiver receives two channel responses from the pilot sequences of each terminal that will be input to the channel estimator. By processing the channel responses the channel estimator will be able to output channel estimations to the modulator.

The estimation procedure will be done regularly to adapt to changes in the channel.

3.1 External Interface

The pilot generator outputs two pilot sequences to the upconverter. The channel estimator receives 28 channel responses from the 28 channels between each of the 2 terminals and each of the 14 transceivers in the MIMO array. Thus, it provides 28 estimates to the modulator. An overview of the external interface of the channel estimator is shown in Figure 3.

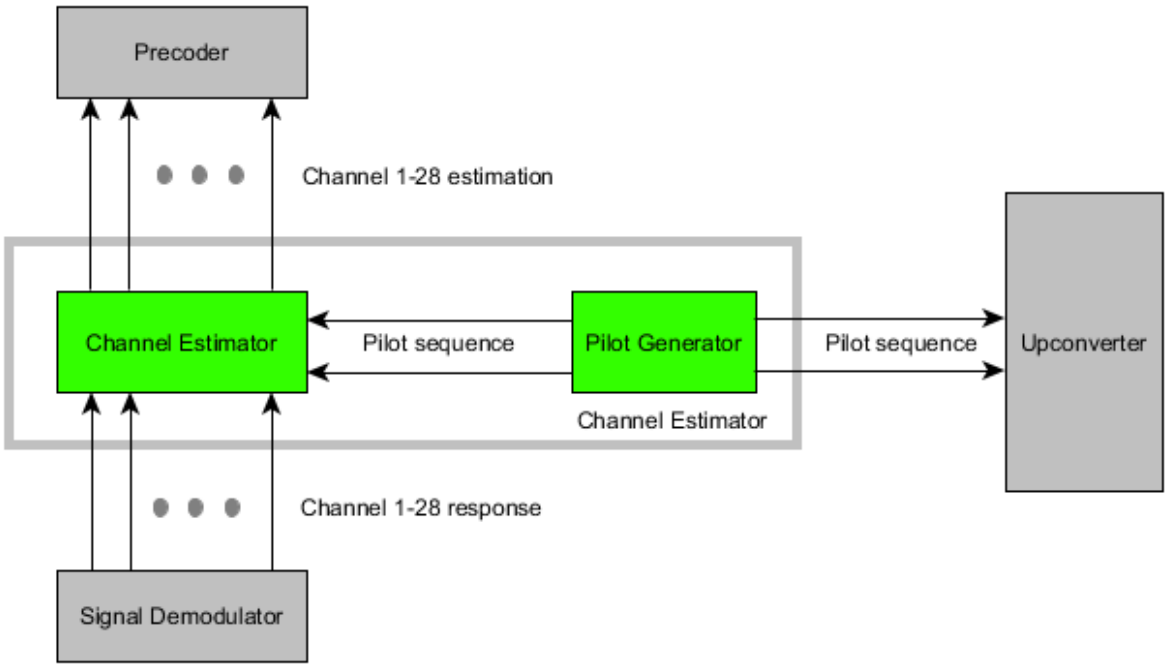

Figure 3: External interface of the channel estimator.

3.2 Mathematical description

Let:

$\boldsymbol{p}_i = \begin{pmatrix} p_i^{(1)} & \cdots & p_i^{(N_p K)} \end{pmatrix}$ be the pilot sequence for the i:th terminal, where N_p is the number of pilot symbols and K is the number of terminals.

$\boldsymbol{P} = \begin{pmatrix} \boldsymbol{p}_1 \\ \vdots \\ \boldsymbol{p}_K \end{pmatrix}$ be the pilot matrix, where K is the number of terminals.

$\boldsymbol{H} = \begin{pmatrix} h_{1,1} & \cdots & h_{1,K} \\ \vdots & \ddots & \vdots \\ h_{M,1} & \cdots & h_{M,K} \end{pmatrix}$ be the channel matrix, where $h_{i,j}$ is the channel impulse response between terminal i and transceiver j, M is the number of array elements and K is the number of terminals.

$\boldsymbol{N}$ be a complex white Gaussian noise matrix.

Then the received matrix $\boldsymbol{Y}$ can be modeled as follows:

$$\boldsymbol{Y} = \boldsymbol{HP} + \boldsymbol{N}$$

3.3 Pilot Generator

Let $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ be the pilot sequence for terminal 1 and 2, respectively. Then $\boldsymbol{p}_1 \cdot \boldsymbol{p}_2 = 0$ must be fulfilled to ensure that the pilot sequences are orthogonal. The easiest way to achieve this is to let

$$\boldsymbol{p}_1 = \begin{pmatrix} p^{(1)} \\ \vdots \\ p^{(N_p)} \\ 0 \\ \vdots \\ 0 \end{pmatrix}^T \text{ and } \boldsymbol{p}_2 = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ p^{(N_p+1)} \\ \vdots \\ p^{(2N_p)} \end{pmatrix}^T$$

A list of inputs and outputs for the pilot generator is displayed in table 4.

Table 4: List of inputs and outputs for the pilot generator.

Input	**Type**
Number of terminals	Integer
Number of pilot symbols per terminal	Integer
Output	**Type**
Matrix containing the pilot sequences (N_pK) for each terminal *(K)*	Matrix ($K\ x\ N_pK$)

3.4 Channel Estimator

The channel estimation will be an estimation of the channel impulse response within the coherence time interval and coherence bandwidth. If the channel can be assumed to be flat fading, the channel impulse response can be represented as a complex number. The LSE algorithm provides an MLE of $\boldsymbol{H}$ according to equation 1.

$$\boldsymbol{H}_{MLE} = \boldsymbol{YP}^H(\boldsymbol{PP}^H)^{-1} \quad (1)$$

Inputs and outputs for the estimation are listed in table 5.

Table 5: List of inputs and outputs for the channel estimator

Input	**Type**
Matrix containing pilot sequences (N_pK) for each terminal *(K)*	Matrix ($K\ x\ N_pK$)
Matrix containing the received pilot sequences (N_pK) from each MIMO-transiever *(M)*	Matrix ($M\ x\ N_pK$)
Output	**Type**
Matrix containing channel estimates for each terminal *(K)* and MIMO transceiver *(M)*.	Matrix *(M x K)*

4 MODULATOR

The Modulator sub-system has two different main tasks. The first is to create signals to be transmitted by the MIMO-transceivers. The second is to demodulate signals received by the two terminals. The MIMO signals will be created by mapping the two bit streams to a predefined constellation and precode the result, with the help of channel state information, into fourteen different signals that are to be transmitted by the MIMO-transceivers.

4.1 External Interface

The input to the modulator sub-system consists of two bit streams from the channel encoder and channel state information, consisting of 28 different channel impulse responses, from the channel estimator. The MIMO signals are output to the I/O drivers. The signals received by the terminals are obtained by the demodulator via the I/O driver. After demodulation and detection the demodulator outputs two bit streams to the channel decoder. A block scheme of the modulator sub-system is presented Figure 4.

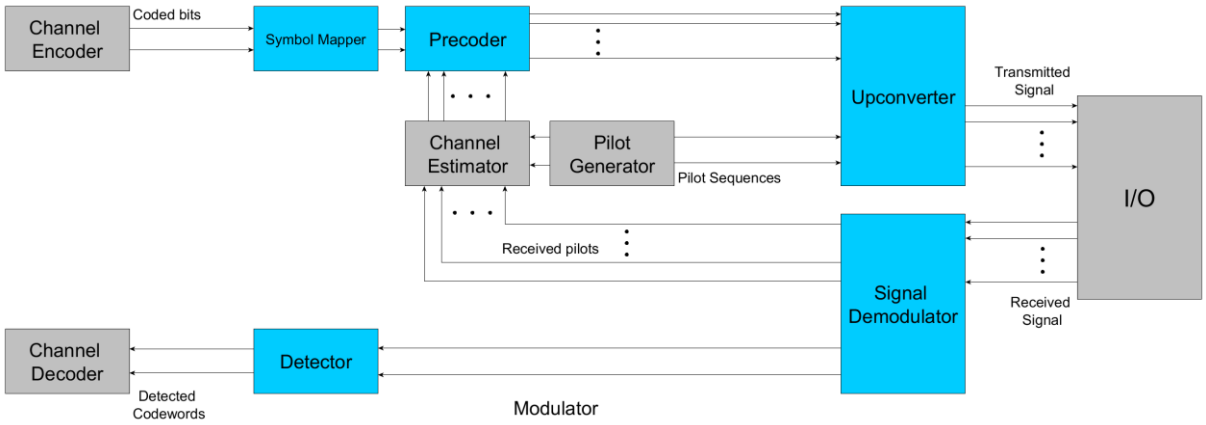

Figure 4: A simple block scheme of the modulator sub-system.

4.2 Mathematical description

Let:

$\boldsymbol{s} = \begin{pmatrix} s_1^{(1)} & \dots & s_1^{(N_s)} \\ \vdots & \ddots & \vdots \\ s_K^{(1)} & \dots & s_K^{(N_s)} \end{pmatrix}$ be the matrix where row i contains all symbols to be transmitted to terminal i.

$\boldsymbol{H} = \begin{pmatrix} h_{1,1} & \cdots & h_{1,K} \\ \vdots & \ddots & \vdots \\ h_{M,1} & \cdots & h_{M,K} \end{pmatrix}$ be the channel matrix for terminal i, where $h_j^{(i)}$ is the channel impulse response between terminal i transceiver j.

4.3 Symbol Mapper

The symbol mapper will map the given bit steam onto the corresponding complex number constellation. A simple BPSK solution will be implemented at first. A more complex solution will be implemented if time permits. Inputs and outputs for the system are listed in table 6.

Table 6: Inputs and outputs of the Symbol Mapper

Input	**Description**
Matrix containing the coded bits (N_c) for each terminal *(K)*	Matrix $(K\ x\ N_c)$
Output	**Description**
Matrix containing complex symbols (N_s) for each terminal *(K)*	Matrix $(K\ x\ N_s)$

4.4 Precoder

The two modulated signals will be precoded by scaling and phase shifting the signals to create fourteen different signals, one signal for each transceiver in the MIMO array, based on the information received by the channel estimator. The precoding will be done by filtering the vector $\boldsymbol{s}$ containing symbols to be transmitted.

$$\boldsymbol{x} = \sqrt{\alpha}\,\boldsymbol{W}\boldsymbol{s} \tag{2}$$

where α is a normalization constant that is calculated as in equation 3.

$$\alpha = \frac{1}{E\{tr(\boldsymbol{W}\boldsymbol{W}^H)\}} \tag{3}$$

Note that this is a theoretical expression as it requires the knowledge of the channel statistics. Since this is not available the expectation will be calculated numerically via experimental data, that is, in the practical implementation α will be an average taken from several estimations of $\boldsymbol{W}$ or by just doing individual estimations for each transmission as in equation 4.

$$\hat{\alpha} = \frac{1}{tr(\boldsymbol{W}\boldsymbol{W}^H)} \tag{4}$$

Three different kinds of filters can be used as $\boldsymbol{W}$, these are: MRT, ZF and MMSE.

Firstly, the MRT algorithm only maximizes the signal gain at the intended user and is close-to-optimal in noise-limited systems, where the inter-user interference is negligible compared to the noise. The MRT algorithm only consists of a filter that will coherently combine the estimated channel responses, given by equation 5.

$$\boldsymbol{W}_{MRT} = \boldsymbol{H}^* \tag{5}$$

This method does not perform well if there is co-channel interference. To take care of that we need to implement ZF or MMSE instead.

The ZF filter only compensates for co-channel interference and neglects thermal noise. For a further explanation of the ZF method read Ngo 2015 (chapter 2). In equation 6 the ZF filter is presented.

$$\boldsymbol{W}_{ZF} = \boldsymbol{H}^*(\boldsymbol{H}^T\boldsymbol{H}^*)^{-1} \tag{6}$$

A more advanced algorithm is the MMSE that minimizes the mean square error. This filter can be viewed as a combination of the MRT and the ZF filter. The filter is equation 7. Once again a further explanation can be read in Ngo 2015 (chapter 2)

$$\boldsymbol{W}_{MMSE} = \boldsymbol{H}^*\left(\boldsymbol{H}^T\boldsymbol{H}^* + \frac{N_0 n_T}{P}\boldsymbol{I}_{n_T}\right)^{-1} \tag{7}$$

Here N_0 is the noise variance, P is the signal power, n_T is the number of terminals and $\boldsymbol{I_{n_T}}$ is an $[n_T\ x\ n_T]$ identity matrix. Then, if the value of $\frac{N_0}{P}$ approaches 0 the MMSE algorithm converge to the MRT algorithm, and if the $\frac{N_0}{P}$ approaches infinity the MMSE algorithm converge to the ZF algorithm.

These linear precoders have low complexity but their efficiency is poor because the noise is colored and in the case of the ZF, the noise is also amplified. But for this project these methods will be sufficient.

Table 7: Inputs and outputs of the precoder

Input	**Type**
Matrix containing complex symbols (N_s) for each terminal *(K)*	Matrix ($K\ x\ N_s$)
Number of MIMO-transceivers	Integer
Matrix containing the impulse response between all MIMO-transceivers *(M)* and all terminals *(K)*	Matrix *(M x K)*
Output	**Type**
Matrix containing precoded complex symbols (N_s) for each MIMO-transceiver *(M)*	Matrix ($M\ x\ N_s$)

4.5 Upconverter

The upconverter is responsible for transforming complex baseband signals into passband signals with a chosen carrier frequency. Inputs and outputs for this system are listed in table 8.

Table 8: Input and output for the upconverter

Input	**Type**
Sample Frequency [Hz]	Double
Carrier Frequency [Hz]	Double
Symbol Rate [Hz]	Double
Matrix containing complex symbols (N_s) for each MIMO-transceiver *(M)* or terminal *(K)*	Matrix ($M\ x\ N_s$) or ($K\ x\ N_s$)
Output	**Type**
Matrix containing real valued passband signals (N_x) for each MIMO-transceiver *(M)* or terminal (*K)*	Matrix ($M\ x\ N_x$) or ($K\ x\ N_x$)

4.6 Signal Demodulator

The signal demodulator is responsible transforming passband signals into baseband. Inputs and outputs for this system are listed in table 9.

Table 9: Input and output for the signal demodulator

Input	**Type**

Sample Frequency [Hz]	Double
Carrier Frequency [Hz]	Double
Symbol Rate [Hz]	Double
Matrix containing real valued passband signals (N_x) for each MIMO-transceiver *(M)*	Matrix ($M\ x\ N_x$)
Output	**Description**
Matrix containing received complex symbols (N_s) for each MIMO-transceiver *(M)* or terminal *(K)*	Matrix ($M\ x\ N_s$) or ($K\ x\ N_s$)

4.7 Detector

The detector will firstly be implemented to do hard decision on symbols. If there is time to implement a turbo code for the channel coding, then the detector will be changed into a soft decision detector. Inputs and outputs for the detector are listed in table 10.

Table 10: Inputs and outputs for the detector

Inputs	**Type**
Matrix containing received complex symbols (N_s) for each terminal (K)	Matrix ($K\ x\ N_s$)
Outputs	**Type**
Matrix containing coded bits (N_c) for each terminal *(K)*	Matrix ($K\ x\ N_c$)

5 CHANNEL CODER

To be able to use the system for demonstration of real data transmission, channel coding shall be used to try to minimize errors in the information bits being transmitted.

The channel coder shall be designed to improve the system, meaning that compared to the system without channel coding it shall preserve the information bit rate while decreasing the information BER. To make this possible, the choice of channel coding parameters must be carefully chosen to fit the type of channel that the rest of the system will present. The channel coder is supposed to input two information bit streams that are to be sent, and output the same number of coded bit streams.

5.1 External Interface

Figure 5 below shows a simple overview of how this sub-system will function in the complete system. The channel coder will be able to do both encoding and decoding. The interface of this will be two functions, one for encoding a number of sequences, and one for decoding a number of sequences. The outputs of the functions will be the decoded/coded sequences.

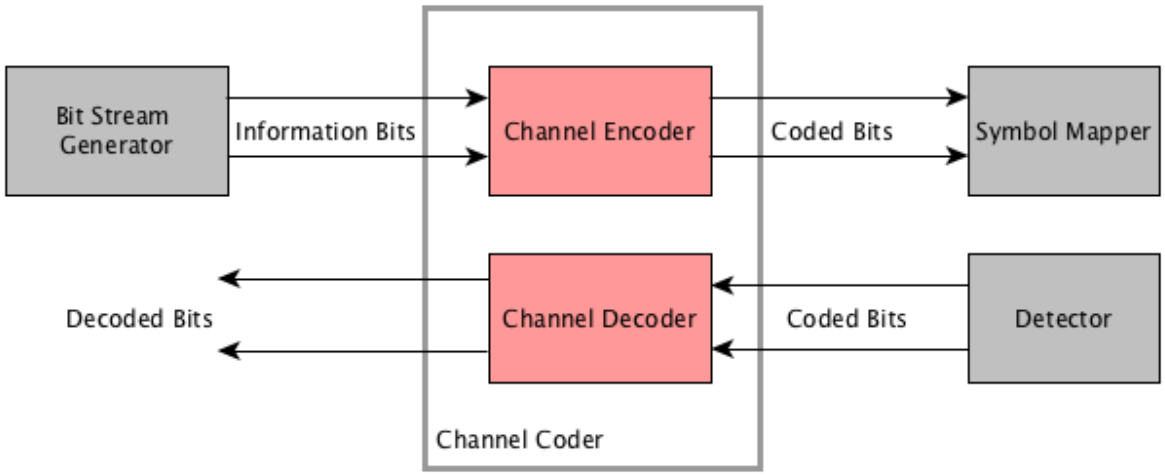

Figure 5: A block scheme of the channel coder

5.2 Channel Encoder

The channel encoder will encode the information bits with a convolutional code. A turbo code encoder will also be implemented if there is time left in the project. The implementation of the channel encoder will be performed using the existing functions in the Communication System Toolbox in Matlab. Table 11 enumerates the inputs and outputs of the encoder.

Table 11: Inputs and outputs of the channel encoder

Input	**Type**
Matrix containing information bits (N_b) for each terminal *(K)*	Matrix of information ($K\ x\ N_b$)
Output	**Type**
Matrix containing coded bits (N_c) for each terminal *(K)*	Matrix ($K\ x\ N_c$)

5.3 Channel decoder

The channel decoder will decode the received code words using firstly the Viterbi algorithm. A turbo decoder will also be created if needed. The implementation of the channel decoder will be performed using the existing functions in the Communication System Toolbox in Matlab. Table 12 lists the inputs and outputs of the Channel decoder.

Table 12: Inputs and outputs of the channel decoder

Input	**Type**
Matrix containing received coded bits (N_c) for each terminal *(K)*	Matrix $(K\ x\ N_c)$
Output	**Type**
Matrix containing received decoded bits (N_b) for each terminal *(K)*	Matrix of information $(K\ x\ N_b)$

REFERENCES

Published references

Svensson, T. & Krysander, C. (2011). Projektmodellen Lips, Upplaga 1:1, Studentlitteratur AB, Lund, ISBN 978-91-44-07525-9

A. J. R. Barry, E. A. Lee, D. G. Messerschmitt, Digital Communication 3rd Ed, Kluwer Academic Publishers 2004.

Electronic references

Stenmark, F. (2014a). User Manual Version 1.1 (PDF) Available: <http://www.isy.liu.se/edu/projekt/kommunikationssystem/2014/documents/UM_1.1.pdf>

Stenmark, F. (2014b). Technical Report Version 1.1 (PDF) Available: <http://www.isy.liu.se/edu/projekt/kommunikationssystem/2014/documents/TR_1.pdf>

Ngo, H, Q. (2015). Massive MIMO: Fundamentals and System Designs, < http://liu.diva-portal.org/smash/get/diva2:772015/FULLTEXT01.pdf>

Unpublished references

(September 7, 2015). The CDIO Project in Communication Systems: Massive Audio Beamforming (PDF)