DESIGN SPECIFICATION

Fredrik Stenmark

Version 1.1

Status

Reviewed	Christoffer Ring	2014-10-07
Approved	Christopher Mollén	2014-10-10

PROJECT IDENTITY

HT 2014
Linköping University, ISY

Name	Role	Phone	E-mail
Anton Eriksson	Hardware Designer	0736143063	anter@592@student.liu.se
Christoffer Ring	Documentation Manager	0738133863	chrri226@student.liu.se
Dimitrios Bakogiannis	Test Manager	0707395508	jbakogia@gmail.com
Fredrik Stenmark	Project Manager	0703823561	frest522@student.liu.se
Magnus Thalén	Chief of Design	0705190977	magth371@student.liu.se

Customer: Christopher Mollén, christopher.mollen@liu.se, 013 – 28 26 43

Examiner: Danyo Danev, danyo@isy.liu.se, 013 – 28 13 35
Supervisor: Antonios Pitarokoilis, antonios.pitarokoilis@liu.se, 013 – 28 13 40

Table of Contents

Document History

Version	Date	Changes	Performed by	Reviewed
1.0	2014-10-07		All group members	Christoffer Ring
1.1	2014-11-11	Corrected minor faults	Anton Eriksson	Christoffer Ring

1 INTRODUCTION

The purpose of this document is to provide a detailed design of the system. The system will demonstrate some of the powers of massive MIMO beamforming. The powers to be shown are that signal energy can be focused to a specific point of interest and that areas around this point will receive much less signal energy.

1.1 MIMO

MIMO (Multiple Input Multiple Output) technology has gained an increasing interest in wireless communication due to its capability to increase the throughput of a communication system without an increased bandwidth or increased transmit power. The technology is based on the use of multiple antennas in both the transmitter and the receiver. By using equal transmit power on all antennas, a higher spectral efficiency is achieved (bits per second per hertz of bandwidth) due to the resulting array gain of the antennas.

1.2 Beamforming

The MIMO technology also allows to direct signal power in a certain direction by altering the phase of the transmitted signal on each of the antennas in order to create constructive interference in a certain point or direction. This enables a transmitter, a base station for example, to direct the communication to a certain receiver without interfering with other receivers.

1.3 The Project

In the course TSKS05 – CDIO Communication Systems – the students are supposed to construct a system that demonstrates the powers, described in the introduction, of massive MIMO beamforming. Using a set of loudspeakers, which can also be used as microphones, the students shall be able to measure sound transmitted from a source and from that sound extract characteristics of the channel between the transmitter and receiver. With this information the loudspeakers shall be able to produce a sound, which will be constructively added at the point where the measured sound originated from. The original project idea was to place a set of four wine glasses next to each other, ringing one of them and letting the system crack this glass, leaving the remaining glasses unaffected. This would be a spectacular way of showing the powers of massive MIMO, which the product aims at demonstrating, since it becomes rather obvious that signal power is focused into one point and suppressed essentially everywhere else. However, since the product is limited to only eight pairs of low powered loudspeakers, the product will not be able to produce enough signal power to crack a wine glass. Therefore the more controlled and secure product setup, described in section 2, will be used.

1.4 Aim and Goal

See section 2.1 in the document Project Plan [2].

1.5 Definitions

Word	Definition
MIMO	Multiple Input Multiple Output
A/D	Analog to Digital
D/A	Digital to Analog
L/M	Loudspeaker and microphone unit
L/M-pair	A pair of loudspeaker and microphone units
OS	Operating System
Subsystem	A part of the whole system

Word	Definition
DB-9	D-sub connector with 9 pins
DB-37	D-sub connector with 37 pins

2 OVERVIEW OF THE SYSTEM

The system consists of two subsystems, Software and Hardware. The user interacts with the Software, which communicates with the Hardware, in order to demonstrate that signal energy can be focused into a very precise point and at the same time be suppressed essentially everywhere else around this point, using massive audio beamforming. The product setup can be seen in Fig. 1. The Software subsystem is hosted on the computer and the Software design is specified in chapter 3. The Hardware subsystem includes everything in the product, which is not included in the Software subsystem, and is further described and designed in chapter 4.

As seen in Fig. 1, a total of eight L/M-pairs will be used in the product. The L/M-pairs are connected to a distribution box, whose task is to provide the correct connections. The distribution box is in turn connected to the A/D and D/A converters, mounted in the computer. When the system operates, seven L/M-pairs will first be used to receive sound data that is sent from the L/M in the point of focus. That is, from one of the L/Ms in the eighth L/M-pair. Then the received sound data will be analyzed and configured in the Software according to chapter 3. The seven L/M-pairs will then focus the configured sound data to the L/M that sent the sound data, i.e. the point of focus. The remaining L/M, of the eighth L/M-pair, will then be used to verify that the signal energy is suppressed in areas around the point of focus. That is, the user will get information concerning the power of the signal in the point of focus and in a point around the point of focus, therefore being able to verify that the signal energy is focused in the point of focus and suppressed around the point of focus. Thereby, some of the powers of massive MIMO beamforming have been shown.

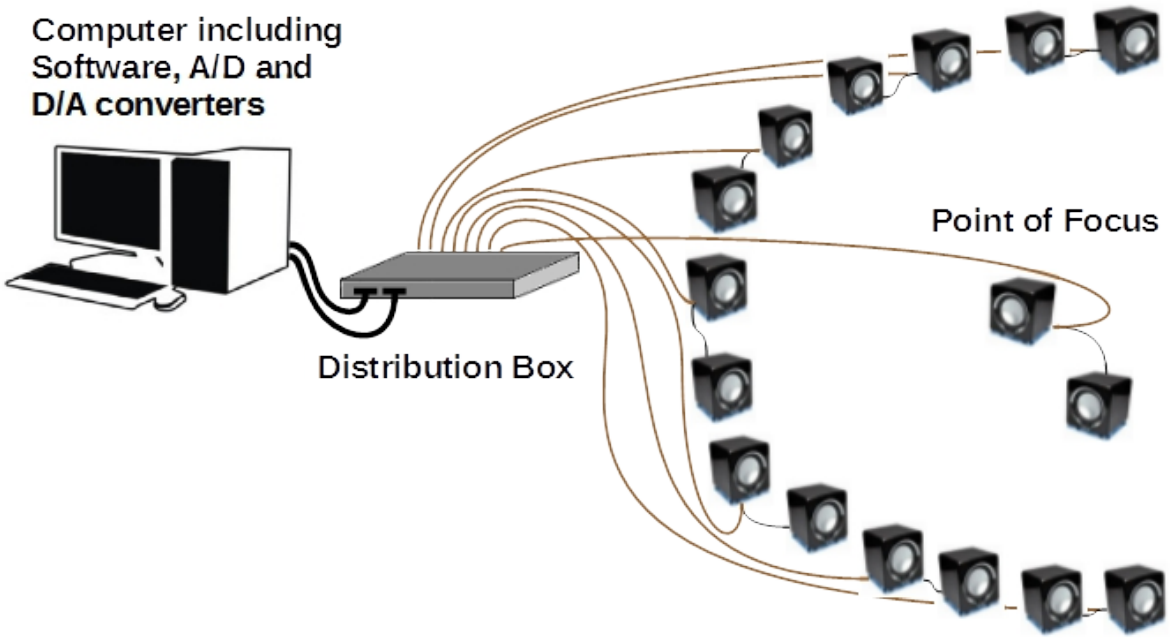

Figure 1: An overview of the system demonstrating that signal energy can be focused into one precise point and at the same time be suppressed essentially everywhere else.

3 SOFTWARE

The Software consists of a user interface, an OS, four software modules and also drivers for the A/D- and D/A converters. The modules are written in MATLAB and are used to take care of all the

computations and will also provide the interface for the user towards the Hardware. The modules and the other parts of the Software are described in sections 3.2 to 3.7.

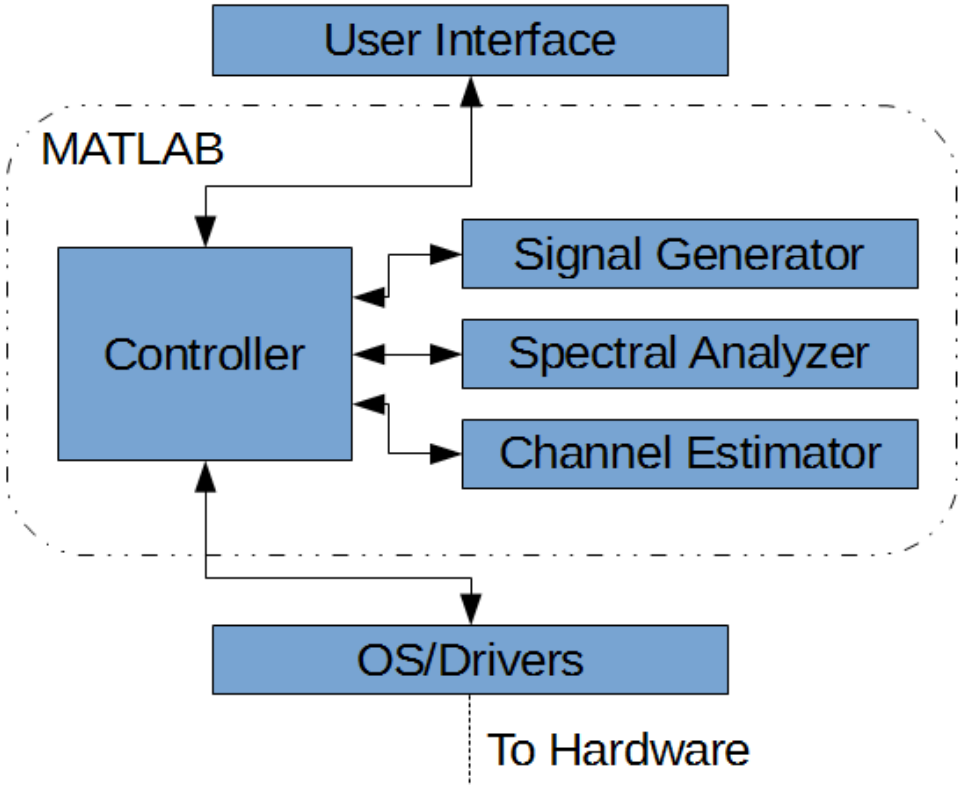

Figure 2: Structure of the Software subsystem.

3.1 General Implementation of Modules

Everything belonging to the different modules (see section 3.3 to 3.6) is MATLAB code in the form of functions or scripts. Both the scripts and the functions shall have a description at the top describing what they do.

Scripts, which are a collection of commands to be performed by MATLAB, have no specific requirements implementation wise other than adhering to good programming style. In this case using a good programming style means that the programmer uses comments in the code and suitable variable names to such an extent that someone who is not aware of the intended function of the program can still follow what is done, though not necessarily how.

Functions on the other hand are parts of the code that take zero or more input parameters, perform a series of commands, and return zero or more output parameters. In addition to adhering to good programming style and having an introductory description describing its purpose, all functions should also specify what data each input and output parameter contains. This should be done immediately after the more general description of the function.

By default, MATLAB stores all numeric variables as double-precision floating-point values. Unless otherwise specified, all output and input to functions will be of this data type.

3.2 User Interface

The User Interface consists of the MATLAB command line interface. The user interacts with the Controller module by calling scripts and functions encapsulated by the Controller module via the command line.

The user interface enables the user to visualize the spectral contents of a recorded signal through the use of MATLAB plots, showing the spectrogram and periodogram of the signal. The user is also able to choose which group of L/M-pairs, out of two, that are operating in which mode, i.e. if the L/Ms belonging to the group are transmitting or receiving. What L/M-pairs belong to what group can be changed through hardware means. The system will also present the characteristics of the signals, which are transmitted, to the user. These characteristics include the amplitude, phase, frequency and power of the signal. Moreover, the system will be able to present what signal is being transmitted over which L/M-pair.

3.2.1 Implementation

There is nothing to implement in the interface part except for installing one of the versions of MATLAB specified at the web page of the provider of the A/D and D/A drivers, www.contec.com/products/daq_util/mldaq.php.

3.3 Controller

The Controller is the main module coordinating the modules described in sections 3.4, 3.5 and 3.6. The main purpose of this module is to centralize the communication between the different modules in order to maintain a manageable structure, enforce loose coupling between the different modules and ease further development. This is done by first gathering sound data, sending these data as input to the other modules and then gathering and using the return values appropriately. The Controller also handles the communication towards the hardware and prepares the signals, using the other modules, for transmission. That is, it ensures that all signals are transmitted synchronously and with the right phase and amplitude.

The controller contains a number of scripts and functions, most of which are used to organize the other modules of the system. All variables of the system are kept in the Controller and are shared between all the scripts. The different variables are shown in table 1.

Table 1: Listing of variables with corresponding descriptions.

Variable	Description
amountLMPairs	The amount of L/M-pairs in the system.
amountLMGroups	The number of groups of L/M-pairs, where all L/M-pairs belonging to the same group are always kept it the same operational mode.
groupAtFocus	Which group of L/M-pairs that is to be used at the point of focus.
carrierFreq	The frequency of the sinusoidal tone that is to be used during the demonstration.
timeDuration	The time duration of the sinusoidal tone that is to be used during the demonstration.
sampleFreq	The sampling frequency that is to be used throughout the system.

3.3.1 Implementation

The primary part of the Controller is a script called Controller_Main that executes the demonstration specified in the Requirement Specification [1], sections 2 and 2.1. The script starts with executing another script, Controller_Init, that sets all the variables that are to be used and initializes the A/D– and D/A converters. When this has been done, Controller_Main waits for user input before proceeding, after which the script sends the specified pilot signal from the point of focus, according to the set variables.

The script then receives the altered signals for each L/M in the receiving group and saves these. For each received signal, the script then analyzes the channel by sending the necessary data to the Channel Estimator. The script then produces the output signals by sending the desired signal and the channel information to the Signal Generator for each L/M. After once again prompting the user for input and having received it, the script synchronously sends the signals to the different L/Ms and proceeds with measuring the data received at the point of focus. When the whole signal has been sent the script plots

a graph for each L/M in the group at the point of focus showing how the input amplitude varied during the signal duration. The pseudo code looks like this:

```
%Description
Controller_Init;
Wait for user input;
signal = constructSignal(carrierFreq, timeDuration, sampleFreq);
synchronize(D/A, A/D);
sendFromPointOfFocus(signal, D/A);
receivedSignals =  ADConverterInput();
for i = 1 : number of L/Ms in group
  channelInformation(i) = ChannelEstimator_Main(sentSignal, receivedSignals(i));
  signalsToSend(i) = SignalGenerator_Main(desiredSignal, channelInformation(i))
Wait for user input;
for i = 1 : number of L/Ms in group
  sendData(signalsToSend(i));
for i = 1 : number of L/Ms in focus group
  plot(dataAtFocus(i))
```

The Controller_Init script sets all the variables necessary to run the demonstration. It works by prompting the user to answer what value each variable mentioned in section 3.3 should have. The user can either input desired value or simply press enter once to use a default values for all variables not set by the user. The pseudo code looks like this:

```
%Description
amountLMPairs = defaultValue;
amountLMGroups = defaultValue;
groupAtFocus = defaultValue;
carrierFreq = defaultValue;
timeDuration = defaultValue;
sampleFreq = defaultValue;
try
  amountLMPairs = askUser('How many L/M-pairs? ');
  amountLMGroups = askUser('How many L/M-groups? ');
  groupAtFocus = askUser('What L/M-group at focus? ');
  carrierFreq = askUser('What signal(tone) frequency? ');
  timeDuration = askUser('What signal(tone) time duration? ');
  sampleFreq = askUser('What samplingFrequency? ');
end
```

3.4 Channel Estimator

The Channel Estimator is used for estimating the channel properties of an individual L/M, based upon the sound data measured by the L/M. The measured signal is provided by the Controller as a

MATLAB vector containing sound data, as a function parameter. The Channel Estimator returns information about the channel's effects on the amplitude and the phase of the sent sound. This is done by comparing the known signal sent for testing purposes with the same signal after having passed through the channel. In order for this to be effective, the noise level has to be minimal meaning that as many noise sources as possible have to be eliminated. One method that will be employed is to isolate the circuits in order to avoid electromagnetic interference. The L/Ms will also be placed on an elastic surface in order to avoid vibrations affecting the measurements.

3.4.1 Implementation

The ChannelEstimator will consist of a main function, ChannelEstimator_Main, which will make use of the subfunctions ChannelEstimator_AmpEst and ChannelEstimator_PhaseEst to calculate the estimated amplitude change and phase difference, respectively.

ChannelEstimator_Main will take two vectors, the sampling frequency, and a flag as input. One of the vectors contains the known pilot signal and the other contains the same signal after being sent through the channel. The flag determines what to plot. Both ChannelEstimator_AmpEst and ChannelEstimator_PhaseEst also take these two vectors as input and return the estimated amplitude change as a factor of the original amplitude, and the difference in phase in radians between the original and altered signal, respectively. ChannelEstimator_PhaseEst also takes the sampling frequency as input. The amplitude change factor and the phase difference make up the output of ChannelEstimator_Main. The pseudo code looks like this:

```
%Description
%Description of input parameters
%Description of output parameters
function [amplitudeChange, phaseDifference] = ChannelEstimator_Main(originalSound, alteredSound,
sampleFreq, flag)
amplitude = ChannelEstimator_AmpEst(originalSound, alteredSound);
phase = ChannelEstimator_PhaseEst(originalSound, alteredSound, sampleFreq);
if(flag == 1)
  plot(originalSound);
if(flag == 2)
  plot(alteredSound); find
if(flag == 3)
  plot(originalSound);
  plot(alteredSound);
```

ChannelEstimator_AmpEst works under the assumption that the change in amplitude is approximately constant over the time period of the signal. It divides both the original signal and the altered signal into several different parts, after which it compares each part of the original signal with the corresponding part of the altered signal. The comparison consists of calculating the quotient between the maximum value of each segment. The function then computes the average quotient and returns it. The pseudo code looks like this:

```
%Description
%Description of input parameters
%Description of output parameters
function [amplitudeChange] = ChannelEstimator_AmpEst(originalSound, alteredSound)
originalAmp = zeros(10,1);
```

```
originalLength = length(originalSound);
for i=1:10
  originalAmp(i) = max(originalSound((i-1)*originalLength/10)+1 : i*originalLength/10));
originalAverage = sum(originalAmp)/10;
alteredAmp = zeros(10,1);
alteredLength = length(alteredSound);
for i=1:10
  alteredAmp(i) = max(alteredSound((i-1)*alteredLength/10)+1 : i*alteredLength/10));
alteredAverage = sum(alteredAmp)/10;
amplitudeChange = alteredAverage/originalAverage;
```

ChannelEstimator_PhaseEst assumes that the input signals are both periodic with period two Pi radians and starts by trying to find the beginning of the measured signal. It does this by cross correlating the two different input signals and finding the maximum amplitude of the resulting signal. The index, where this maximum amplitude is found, corresponds to the delay. Using the sampling frequency of the vectors, this index offset is then converted to an offset in phase, which is then returned. The pseudo code looks like this:

```
%Description
%Description of input parameters
%Description of output parameters
function [phaseDifference] = ChannelEstimator_PhaseEst(originalSound, alteredSound, sampleFreq)
corrSignal = xcorr(originalSound, alteredSound);
indexOffset = find(max(corrsignal)==corrSignal);
phaseDifference = indexOffset % wavelength converted to amount of index steps;
phaseDifference = phaseDifference*indexToRadiansFactor;
```

3.5 Spectral Analyzer

The purpose of the Spectral Analyzer, which is a single function, is to calculate the main frequency component of a signal. The input parameters are a sampled vector containing sound data, the sampling frequency, and a flag determining what to plot. The only output is the frequency, which is returned to the caller of this function.

3.5.1 Implementation

The Spectral Analyzer consists of one function called SpectralAnalyzer_Main. This operation is done by first calculating the discrete Fourier transform of the vector containing the signal, finding the index of the maximum value of the vector, converting the index to its corresponding frequency and then returning this frequency. The flag determines whether the Spectral Analyzer will also plot the periodogram and/or the spectrogram. The periodogram, which is an estimate of the spectral density function, will be acquired by first calculating the autocorrelation and then extracting the Fourier transform through a Fast Fourier Transform (FFT). The spectrogram will be acquired through a FFT of the original sound data. The plotting itself will be done with the help of the available plot function in MATLAB. The pseudo code detailing the implementation looks like this:

```
%Description
%Description of input parameters
%Description of output parameters
```

```
function [freq] = SpectralAnalyzer_Main(signal, sampleFreq, flag)
transform = fft(signal);
if(flag == (1 || 3))
  xlabel('freq range');
  ylabel('amp');
  plot(transform);
if(flag == (2 || 3))
  r = acf(signal);
  R = fft(r);
  xlabel('freq range');
  ylabel('amp');
  plot(R);
m = max(transform);
maxIndex = find(transform == m);
freq = maxIndex*correctConversionFromIndexToFrequency;
```

3.6 Signal Generator

The Signal Generator generates a vector of samples constituting the signal to be sent to an L/M. The input consists of a vector describing the desired signal at the focus point and the estimated changes to amplitude and phase imposed by the channel. The output is a vector containing the sampled sound data to be sent to an L/M.

To achieve this, the Signal Generator calculates the signal characteristics needed for transmission, based upon the information generated by the Channel Estimator.

3.6.1 Implementation

The signal generator consists of only one function, SignalGenerator_Main, that takes a vector of sound data, the amplitude change imposed by the channel, the phase difference imposed by the channel and the sampling frequency. It returns a vector of sound data that has the same dimensions as the one in the input. The output vector consists of the same values as the input vector but shifted and scaled to counter the changes that the channel will likely apply to the signal when transmitted, as detailed by the amplitude change factor and the phase difference.

The amplitude of the signal is scaled by the inverse of the scaling of the channel. The phase of the signal is shifted the same amount as the phase shift of the channel but in the opposite direction, unless the phase shift is negative, in which case it is instead additionally shifted in the same direction so that the total shift equals a full period of the signal. The pseudo code looks like this:

```
%Description
%Description of input parameters
%Description of output parameters
function [outputSignal] = SignalGenerator_Main(desiredSignal, amp, phase, sampleFreq)
%Scale vector
outputSignal = desiredSignal.*(1/amp);
%Shift vector
if(phase < 0)
```

```
  phase = 2pi – phase;
indexShift = phase shift in terms of index using sampleFreq etc;
outputSignal(indexShift : end) = outputSignal(1 : end – indexShift – 1);
outputSignal(1: indexShift – 1) = zeros(indexShift – 1, 1);
```

3.7 OS/Drivers

The OS and the drivers constitute the interface between the MATLAB code and the Hardware. The installed OS is Microsoft Windows 7. The different drivers are the drivers for the A/D and D/A converters. Additionally there is also a required add on for MATLAB called Data Acquisition Toolbox (DAQ) as well as a MATLAB library called MATLAB-compliant Data Acquisition Library (ML-DAQ).

The OS is provided by the department of Electrical Engineering, ISY. The drivers for the A/D and D/A converters are provided by the manufacturer of the D/A and A/D converters, Contec, at their web page www.contec.com. The DAQ is included in the student license for MATLAB, which is to be used and the ML-DAQ is provided for free by Contec at their web page. Installation of the drivers is done according to the instructions enclosed with the drivers. The DAQ is installed automatically along with MATLAB. The ML-DAQ is installed using the available DLL file.

4 HARDWARE

The Hardware consists of a number of entities. These are the Computer, D/A converter, A/D converter, distribution box, amplification circuits and L/M-pairs. Fig. 3 presents the architecture of the Hardware. The following sections of this chapter describes these entities in more detail.

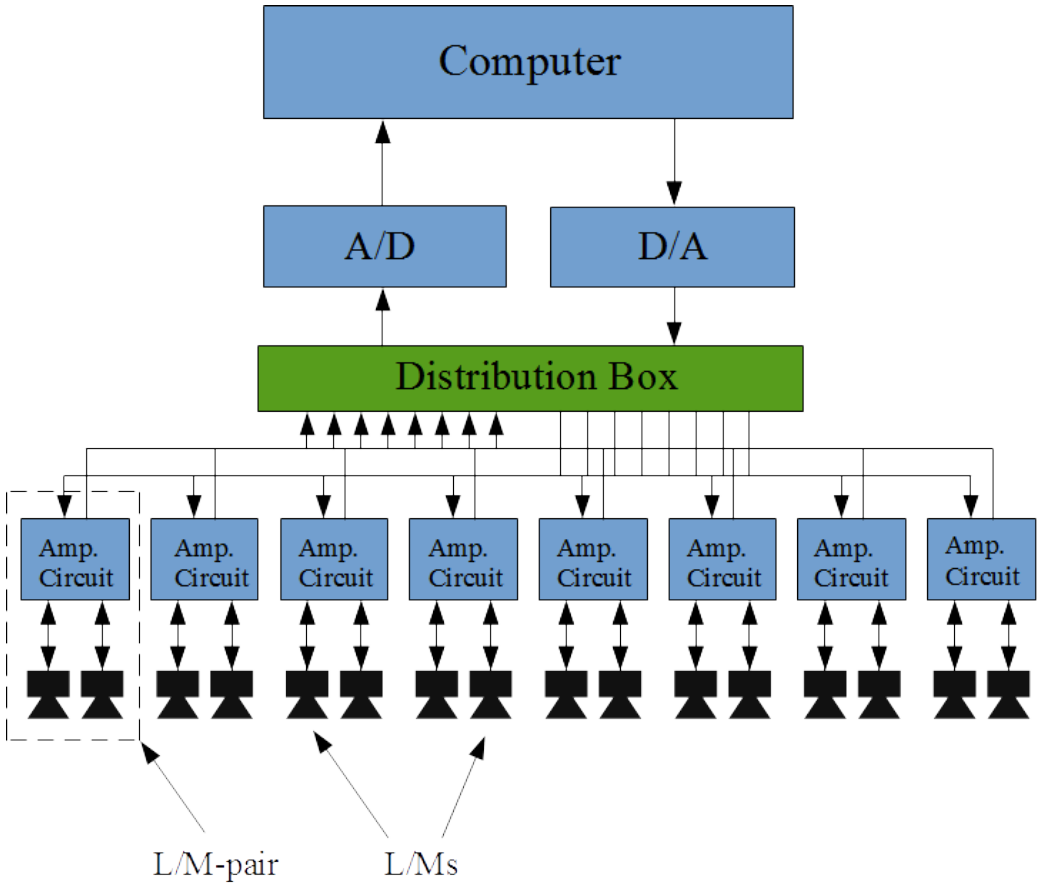

Figure 3: Structure of the Hardware subsystem.

4.1 Computer

The Computer is the platform where all the Software is operating. The Computer communicates with the A/D and D/A converters which are connected to the PCI ports of the computer.

The Computer is a stationary computer provided by ISY.

4.2 A/D and D/A Converters

The A/D and D/A converters are, as stated in section 4.1, connected to the PCI ports of the Computer and serve as a communication channel between the Computer and the L/M-pairs. There is one A/D converter and one D/A converter.

The converters are provided by ISY. The A/D converter is of type Contec AD12-64 (PCI) and the D/A converter is of type Contec DA12-16 (PCI). Both the A/D and the D/A converters are connected to a customized distribution box, which design is shown in section 4.4.

The output from the D/A converter can be set as bipolar with voltage level ±5V. There is thus no need for a level conversion circuit at the input of the L/M-circuit, since the amplification circuit of the speakers requires a signal with mean zero. The level of the signal must however be controlled by the Software in order to not damage the amplification circuit and must be limited by a factor ½.

The input level for the A/D converter will be set as bipolar in the range of ±10V. The A/D converter also has four digital output pins for TTL level signals (Transistor-Transistor-Logic), which are intended for controlling the relay switches. One of the digital pins controls one group of L/M-pairs

whilst another one controls the second group. A more detailed description of these connections is given in section 4.4.1.

4.3 L/M

The L/M works both as a microphone and as a loudspeaker and contains an original amplification circuit for outgoing signals and a detection circuit for ingoing signals, designed by Mikael Olofsson. The detection circuit amplifies the signals from the loudspeaker in order to use it as microphone, it also contains the mode switching circuitry.

The power supply to each L/M-pair is provided by a USB cable. These USB cables are connected to a wall socket via a USB adapter.

The entities of the L/M-pairs will be described in more detail in sections 4.3.1 and 4.3.2.

4.3.1 Maxxtro Mini Speaker 4 W

The loudspeakers used in the product are “Maxxtro mini speaker 4 W”. Each L/M-pair will consist of one modified pair of loudspeakers. The loudspeakers are active and driven by 5V DC, which is provided by a USB connection. Each pair of loudspeakers has one master unit and one slave unit. The master unit contains the original amplifier circuit, which is used to amplify outgoing audio signals. For each loudspeaker pair, every master unit will also be fitted with a detection board, see section 4.3.2, which is used to amplify ingoing audio signals. Further on, in order to enable communication with the Computer, these speakers will also be fitted with a DB-9, see Fig. 4. A total of 8 loudspeaker pairs will be used in the product.

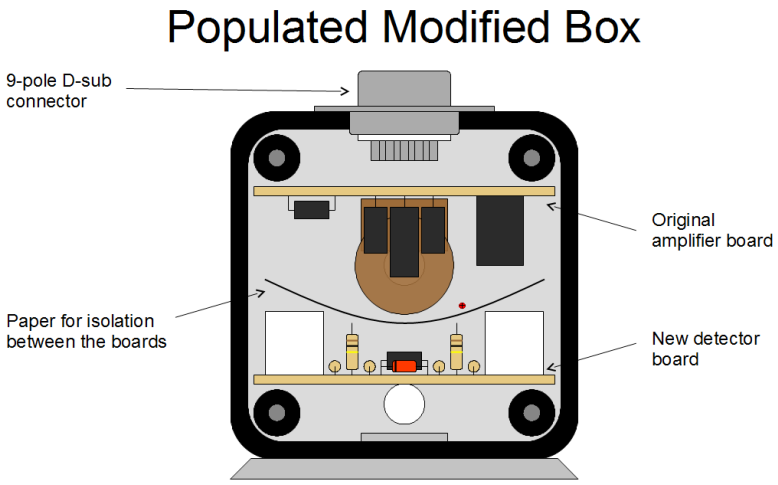

Figure 4: Overview of the L/M master unit [3].

4.3.2 Detection board

The detection board, see Fig. 5, designed by Mikael Olofsson, will be used to amplify ingoing audio signals when the L/M operates as a microphone. The detection board uses a differential-in-differential-out amplifier with a voltage gain of 23dB. The amplified outputs from this detection board will be connected to a collection board, see section 4.4.2, which is placed in the distribution box, see section 4.4, and contains differential amplifiers that each has a voltage gain of 10dB. This means that we have a total voltage gain of 33dB for each L/M, when using them as microphones. The connection between the detection board and the distribution box is realized via a cable that is connected to the DB-9 on the L/M. The components needed for one detection board is listed in table 2.

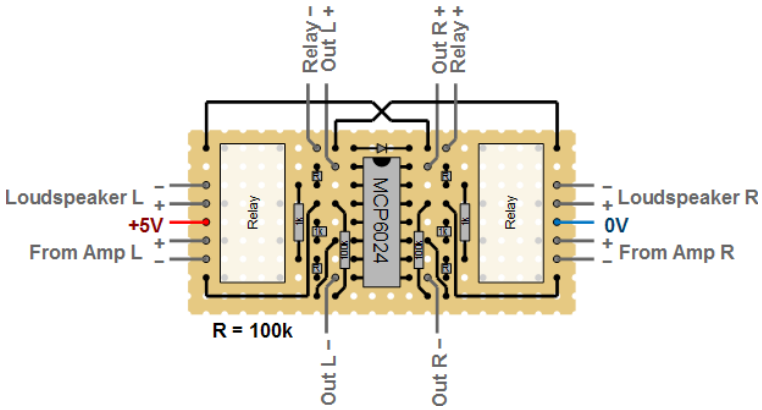

Figure 5: Design of the detection board [3].

Table 2: Listing of components required for one detection board.

Component type	Value	Product number	# of units needed	Stock number (ELFA)
Resistor	1k Ohm	-	4	-
Resistor	100k Ohm	-	6	-
Relay	5VDC, 167 Ohm	HFD27F/005-H 051	2	37-157-20
OP-Amp	Quad	MCP6024-I/P	1	73-786-63
Capacitor	1µF	-	1	-
Diode	75V	1N914	1	70-190-50

4.4 Distribution Box

Each L/M needs to be connected to both the A/D converter and the D/A converter to enable communication both from and to the Computer. This is done by gathering all connections in a customized distribution box which layout is shown in Fig. 6. The distribution box contains three circuit boards, one for ingoing signals, one for outgoing signals and the collection board designed by Mikael Olofsson. The distribution box has a total of 11 cable connections. Eight of these are connections for the L/M-pairs, and then there are one connection each for the A/D and D/A converters. Each L/M-pair is connected to the box with a DB-9 and the D/A converter is connected with a DB-37. The distribution box also has a connection for power supply, which is provided by a power supply unit from a stationary computer. The switches in Fig.6 are physical switches, controlling which control group an L/M-pair belongs to. The concept of control groups are described in section 4.4.1.

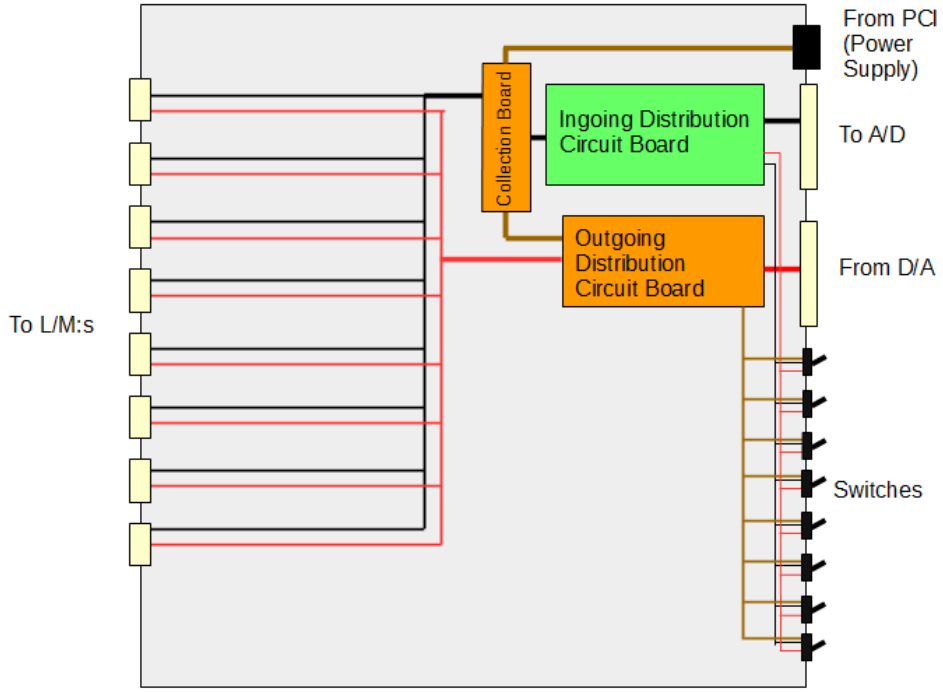

Figure 6: The layout of the distribution box.

4.4.1 Outgoing Distribution Circuit Board

The layout of the circuit board for outgoing signals from the D/A converter and digital output from the A/D converter is shown in Fig. 7.

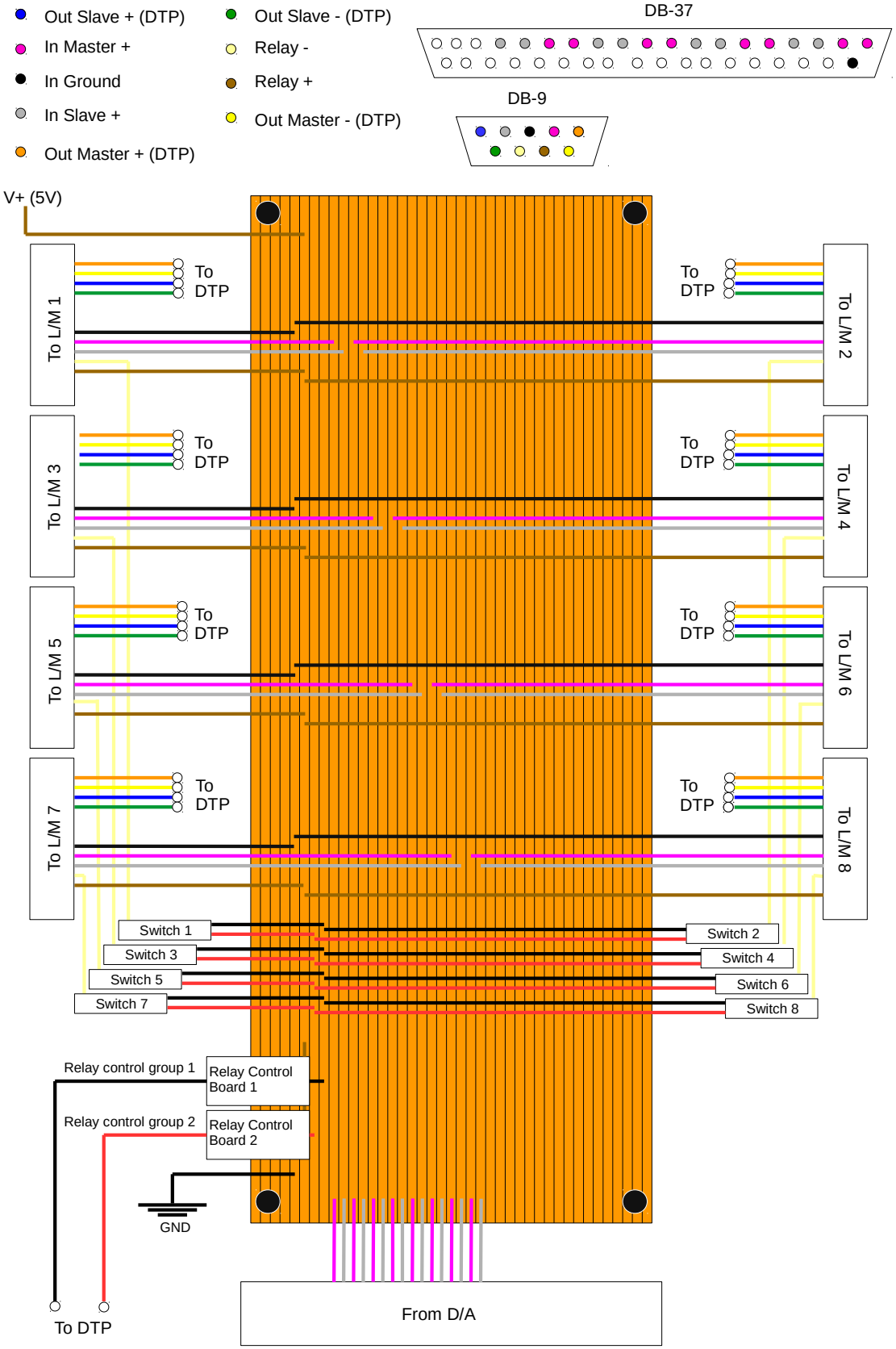

Figure 7: The layout of the outgoing distribution circuit board.

The DB-9 connectors constitute the connection to the L/M-pairs and are divided into input and output from the L/M-pairs with four pins each, and ground. The input to the L/M has four connections which are signal input and relay control, two connections each. The signal input is channeled directly from the D/A converter via the DB-37 in the distribution box. The relay controls from the different L/M-pairs are divided into two control groups and each L/M pair is switched to either of the two using a physical switch. Each control group is connected to the relay control board, which input is wired onto the ingoing distribution circuit board, see section 4.4.3, where the digital outputs from the A/D converter are directed from.

The relay control board is included in the outgoing distribution circuit board and consists of two transistor configurations, one for each control group. This constitutes the relay control board in the system overview schematic, see appendix A. Each transistor configuration on the relay control board consists of two NPN transistors connected in line with a resistor as in Fig. 8, resulting in a configuration similar to a Darlington transistor. The purpose of this circuit is to make sure that the current that runs trough the relays in an L/M-pair is high enough. The transistors are therefore chosen so that the second transistor in line is capable of conducting the maximum current of 470mA, which is the case of having all 16 relays switched to the same control group. The first transistor supplies the base current to the second transistor high enough to ensure saturation without stressing the digital output of the A/D converter.

The components needed for a relay control board are listed in table 3.

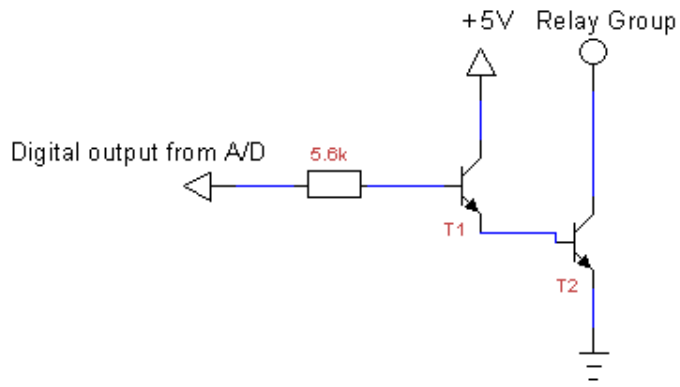

Figure 8: Schematic of one, out of two, transistor configurations that are used in the relay control board.

Table 3: Listing of components required for one relay control board.

Component type	Value	Product number	# of units needed	Stock number (ELFA)
Resistor	5.6k Ohm	-	2	-
Transistor	gain = 250	BC337-40	2	71-014-59
Transistor	gain = 15	BSX47-10-T	2	71-301-48

4.4.2 Collection Board

The collection board consists of differential amplifier circuits, each with a gain of 10dB, see Fig. 9. Each L/M-pair will be connected to one amplifier circuit on the collection board. This means that there are 16 amplifier circuits on the collection board. The collection board is placed in the distribution box, see section 4.4. The components needed for the collection board is listed in table 4.

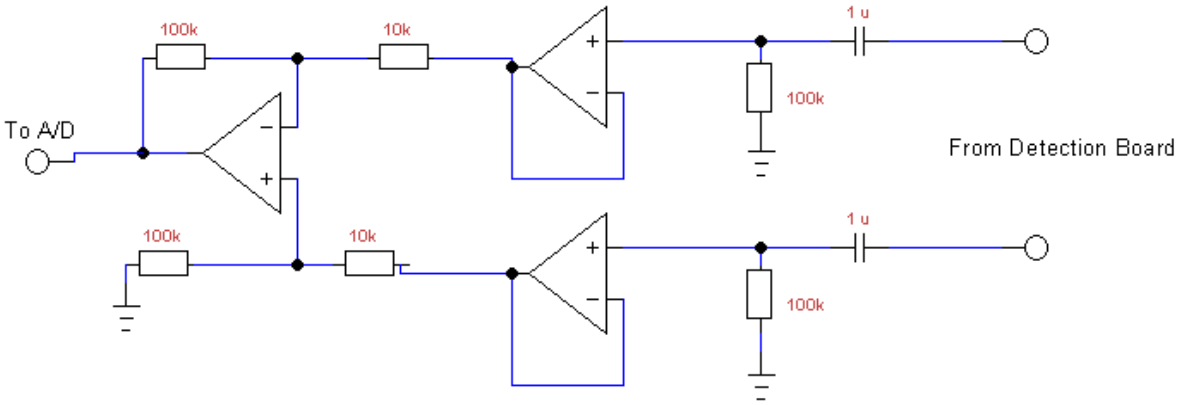

Figure 9: Schematic of one, out of sixteen, amplifier circuits that are used in the collection board.

Table 4: Listing of components required for one collection board.

Component type	Value	Product number	# of units needed	Stock number (ELFA)
Resistor	10k Ohm	-	32	-
Resistor	100k Ohm	-	64	-
Capacitor	1µF	-	32	-
OP-Amp	Quad	TL074CN	12	73-117-56

4.4.3 Ingoing Distribution Circuit Board

The circuit board used mainly for ingoing signals is a Contec DTP-64 screw terminal, which allows for connecting cables without soldering. The screw terminal is mounted inside the distribution box and connected to the A/D converter via a PCA cable with 96 pins. The mapping of the signals is shown in Fig. 10.

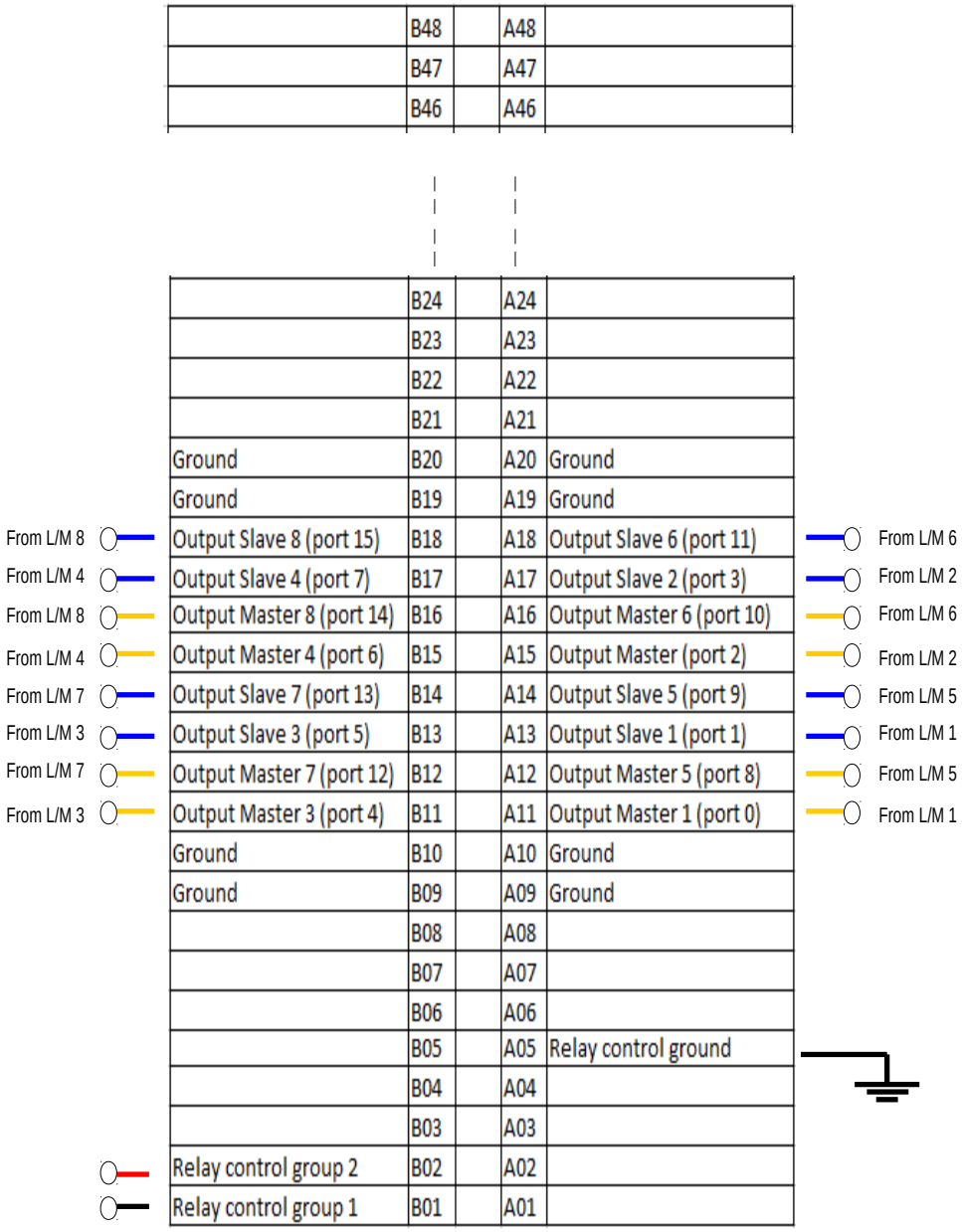

Figure 10: Plan of the ingoing distribution circuit board. The port numbers correspond to the ports to be addressed in the Software.

REFERENCES

Internal Documents

[1] Stenmark, Fredrik (2014), *Requirement Specification*. Massive Audio Beamforming, Linköping University

[2] Stenmark, Fredrik (2014), *Project Plan*. Massive Audio Beamforming, Linköping University

[3] Olofsson, Mikael (2014), *HT-design.pptx.* Massive Audio Beamforming, Linköping University

A SYSTEM OVERVIEW SCHEMATIC