PRO-Q2 Final Design Report

Daan Conijn, 13023217
Andrew Lau, 13058339
Kevin Oei,13090062
Koen van Vliet, 13093053
Group 1

EQ2.a
EQ2.c

June 12th, 2015

Contents

1 Summary

Section 2 gives an introduction of the project PRO-Q2 of team 1.

Section 3 explains the problem description regarding the design of the active loudspeaker in detail. The specifications and the preconditions are also presented in section 3.

Section 4 shows the design overview of the active loudspeaker. Schematics of the preamplifier, power supply, power amplifier and the DCU are presented in this section. The values of the components are also presented in the schematics.

Section 5 gives an in-depth analysis of the preamplifiers regarding the calculations, simulations and measurements.

Section 6 explains the design of the power supply in detail. It explains what choices were made in the design of the power supply

Section 7 explains what choices were made in designing the DCU, what code the DCU consists of and how the DCU functions.

Section 8 gives the results of the test procedure and how the tests were done.

Section 9 gives the total cost of the system. The prices and the types of the components are presentend in this section.

Section 10 gives the conclusion and recommendations of the project as a whole.

2 Introduction

In the second year of the Electrical Engineering course at The Hague University of Applied Science, project PRO-Q2 marks the end of the primary programme. All the specific knowledge and skills needed in previous projects come together in this one project with a duration of one semester. Students are presented with the challenge of designing a complete speaker system with digital enhancements. This report will offer the reader a detailed look into all of the design aspects realized by team 1 of PRO-Q2, as well as test results and an overview of the total costs of the required materials. After reading this report, one should be able to assemble the functional product as described in the report.

3 The project problem, specifications and preconditions

The project problem introduces the fictional company BARK, a small yet well-known manufacturer of loudspeakers [1]. For some time, BARK has contemplated the development of a series of active speakers using built-in power amplifiers. The market price of these types of speakers have shown a significant reduction in the recent years, thus making it an interesting move for the company. Speakers with this setup offer a range of great advantages, including the avoidance of large speaker cables, the possibility of using cheaper electronic crossover circuits and a straight-forward design of the circuits.

It is likely that buyers of these active speakers also have the intention of buying a high-end stereo preamplifier. These types of preamplifiers are known to be expensive and uncommon. For this reason, BARK developed a new preamplifier to be built into the speaker system. The specifications of the preamplifier have already been established by a previous team.

3.1 The problem

In light of the recent digitization of every electronic product, BARK decided that it also wants the speaker and its electronics to be controlled by a digital system of some sorts. However, being a loudspeaker manufacturer, their knowledge of analogue and digital electronic systems is somewhat limited. A series of researches are to be conducted in order to gain the required knowledge and skills in these fields. The project team is to design an active loudspeaker system and realise a demonstration model according to the specifications and preconditions given. By the end of the project, the live demonstration should prove the correct operation. Documentation and a cost calculation are also to be provided.

3.2 Specifications and preconditions

The specifications and preconditions given by BARK encompass both the analogue as well as the digital part of the to-be-designed system. The following components are described:

- Preamplifier
- Power amplifier
- Power supply
- Digital part

3.2.1 Preamplifier

The configuration of the preamplifier has already been partially determined. The values of the components have to be calculated and additional components have to be implemented in order to make the design of the preamplifier meet the required specifications.

The specifications are as follows:

Overall specifications preamplifier Balance control in flat position, tone (bass and treble) control switched off, volume control at maximum.

1. Input impedance: minimum 100 kΩ
2. Output impedance: maximum 100 Ω
3. Sensitivity : 2 V output voltage in R_L = 20 kΩ at 200 mV input
4. Frequency range: 2 Hz to 400 kHz (-3 dB) at V_{out} = 500 mV, R_L = 20 kΩ
5. Output voltage: at least 7 V sinusoidal with R_L = 20 kΩ , f = 1 kHz

Balance control specifications

1. Range $\pm$6 dB (within $\pm$0.5 dB) difference between right and left channels
2. In flat position +2 dB (within $\pm$0.5 dB)
3. Approximate logarithmic control function
4. No DC current through the potentiometer nor the wiper

Bass and treble control specifications

1. Treble: Cut-off frequency adjustable between 2.5 and 10 kHz. Maximum boost or attenuation 10 dB ($\pm$1 dB) at 2.5 kHz. From 100 kHz and up the transfer function must approach value 1 (0 dB).
2. Bass: Cut-off frequency adjustable between 100 Hz and 400 Hz. Maximum boost or attenuation 10 dB ($\pm$1 dB) at 400 Hz. From 10 Hz and below the transfer function must approach value 1 (0 dB).

3. Approximate logarithmic control functions for cut-off frequencies and boost / attenuation adjustments

4. No DC currents through potentiometers nor wipers

5. **Option**: tone controls are switched on and off without any clicking noises from the speakers

Volume control specifications

1. Approximately logarithmic control function

2. No DC current through the potentiometer nor the wiper

The preconditions are as follows:

1. The potentiometers for balance, bass, treble and volume control are 10 kΩ linear.

2. The potentiometers for the adjustment of the bass / treble cut-off frequencies are 10 kΩ logarithmic.

3. All the preamplifier's opamps are low noise NE5532 type. The NE5532 is a dual opamp version of the NE5534. Both Multisim and Pspice contain suitable models for the NE5532 and the NE5534.

4. The simulation results of the sub circuits and the preamplifier are represented by the transfer function (amplitude and phase characteristics). These simulation results must be compared with the transfer function gained by measurement. Draw your conclusions.

3.2.2 Power amplifier specifications

1. Input impedance of at least 50 kΩ

2. Output power: 15 W sine in $R_{load} = 8\ \Omega$ at 1 kHz

3. Frequency range: 10 Hz to 100 kHz (−3 dB) at $P_{load} = 0.5$ W in 8 Ω

For simplicity of design, the power amplifier should be a monolithic component.

3.2.3 Power supply

The power supply must supply all rated voltages and current in the system as to be determined by the project team. The power supply should be constructed using linear regulators. The projected efficiency should be 80% or better.

3.2.4 Digital part

The project team is to determine and develop the functionality of the digital part of the system. BARK mainly deals with analogue electronics, and so the project team is asked to help develop attractive features that are to be implemented within the project's duration. The features should be discussed with the project coach beforehand. The system will consist of at least:

- An ATmega32 microcontroller or similar;
- Some kind of interaction with the analogue part of the system.

4 Design overview

As already hinted, the system primarily consists of four subsystems. They are the preamplifier, power amplfier, power supply and the digital component (referred to as 'DCU' from here on, short for 'Digital Control Unit'). While the design for the preamplifier is already predetermined for the most part, all other parts have to be designed from scratch. An overview of the system's design will be presented in this section, with minor to no additional clarifications on the design itself. The sections that follow will offer a detailed look into the analysis/design of the parts.

4.1 Preamplifier

The design of the preamplifier can be divided in the following parts:

- Balance control
- Tone (bass and treble) control
- Volume control

These parts will be looked at in greater detail separately, figure 1 shows a schematic overview of the entire preamplifier to show the connection between the different parts.

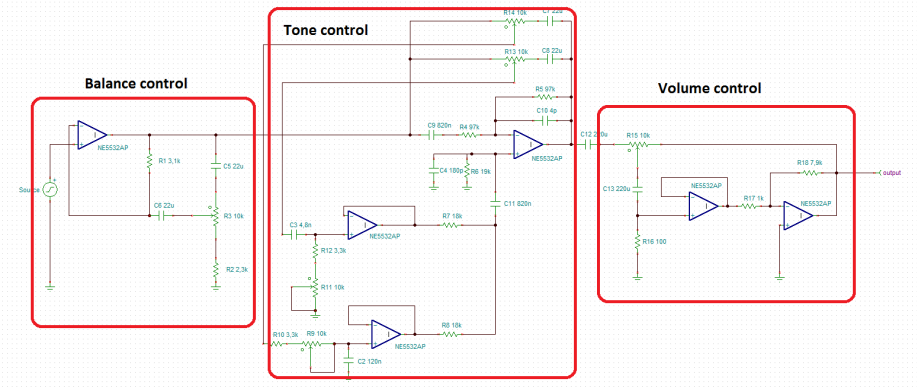

Figure 1: Schematic preamplifier

All parts of the preamplifier have been realized on prototyping boards.

A capacitor that filters out the AC signal removing the noise and provides a DC signal is known as a bypass capacitor. The capacitors connected at V_{cc} and V_{ee} in figure 2 is a bypass capacitor bypassing AC noise and allowing pure DC signal to pass through the component.

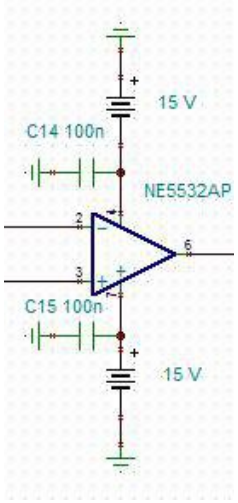

Figure 2: Schematic preamplifier

4.1.1 Balance control

The schematic of the balance control is as shown in figure 3. It function is to be able change the difference between the right an left channels. Thus if a stereo configuration is used then the balance of the output can be adjusted. Since there is only one channel in our configuration the balance control will function as an volume control with a range of 0 to 6dB. The digital potentiometer on the balance control is adjustable by using the DCU.

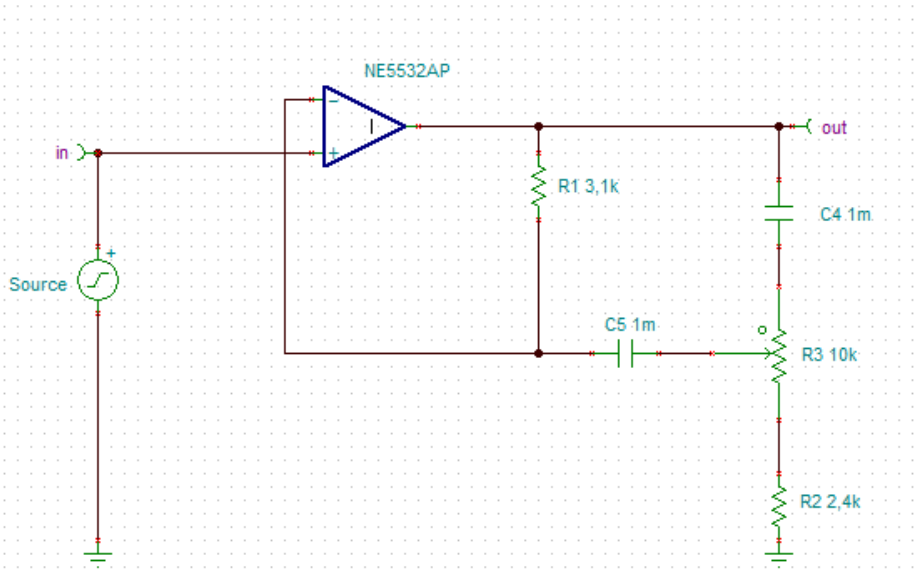

Figure 3: Schematic of balance control

4.1.2 Tone control

The schematic of the Tone control is as shown in figure 4. The tone control will filter out the frequencies who are not in a 2Hz-400kHz range. Between this range there are two potentiometers who are used to adjust the Bandwidth of the Bass and Treble. Bass has a range from 10Hz through an adjustable Cut-off frequency between 100Hz-400Hz. Treble has a range from an adjustable Cut-off frequency between 2.5kHz-10kHz through 100kHz. Within these ranges the signal can be boosted or attenuated by 10dB. All the potentiometers can be controlled using the DCU.

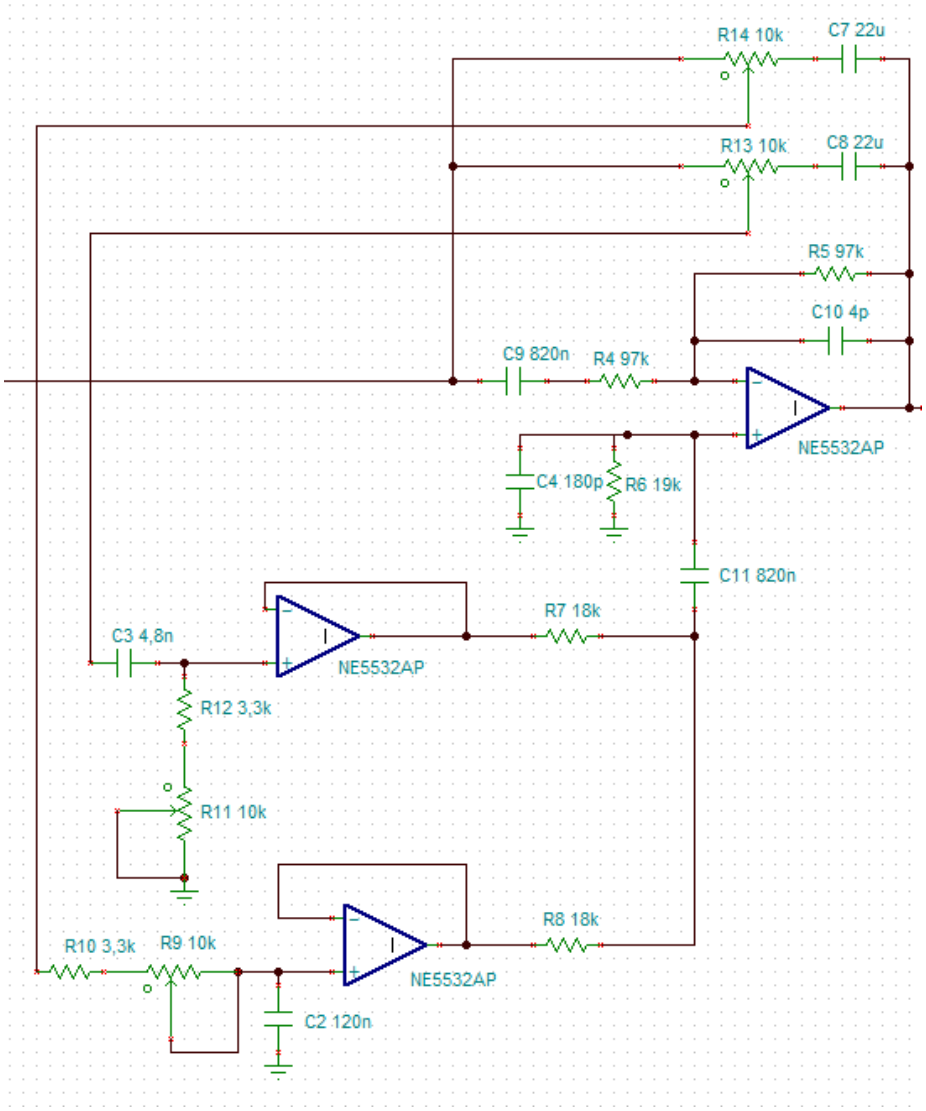

Figure 4: Schematic of tone control

4.1.3 Volume control

The schematic of the Volume control is as shown in figure 5. The volume control is used to boost or attenuat the signal as a whole. By adjusting the digital potetiometer the signal can be boosted or attenuated within an range of -24dB through 18dB.

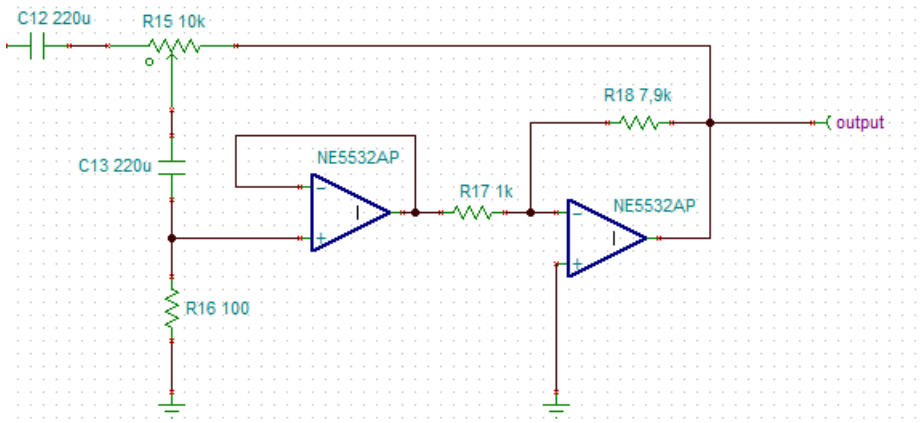

Figure 5: Schematic of volume control

4.2 Power amplifier

The schematic of the power amplifier design is as shown in figure 6.

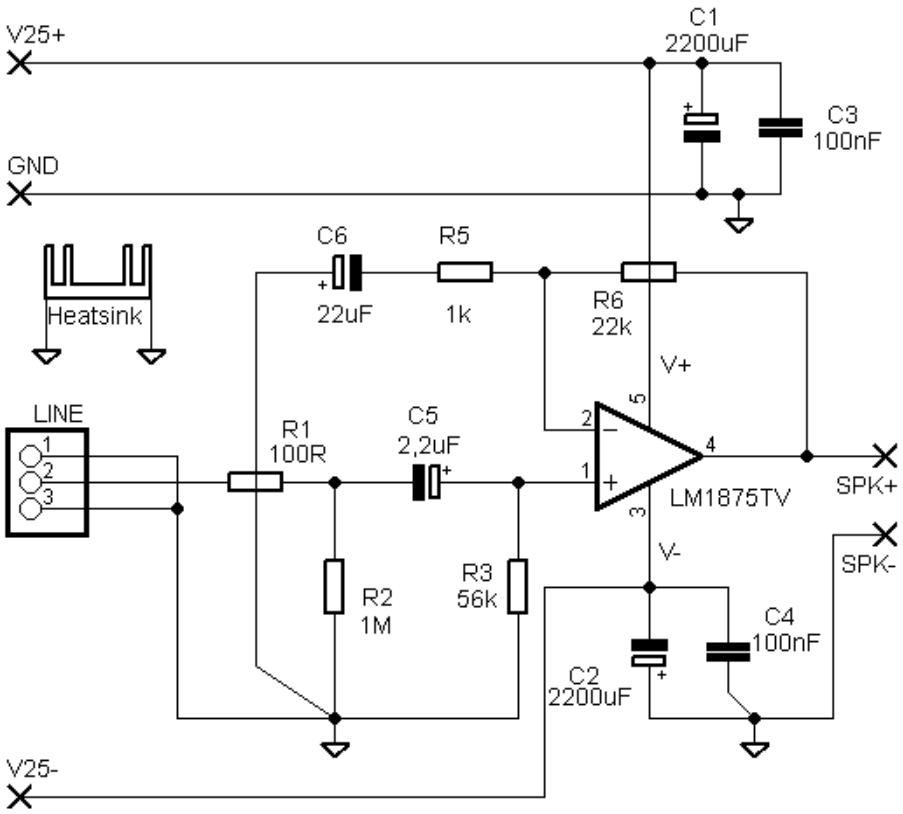

Figure 6: Schematic of power amplifier

Note that pins 1 and 3 of the line-in (named 'LINE' in the schematic) are both connected to ground. This allows for plugging of the mating connector without having to worry about the connector being plugged in the wrong way around. It is also worth noting that a 100 Ω resistance is connected in series with the input to protect the amplifier against overvoltage. The added resistance allows the clamping diodes inside the amplifier to deal with any excess voltage.

The PCB design of the power amplifier is as shown in figure 7.

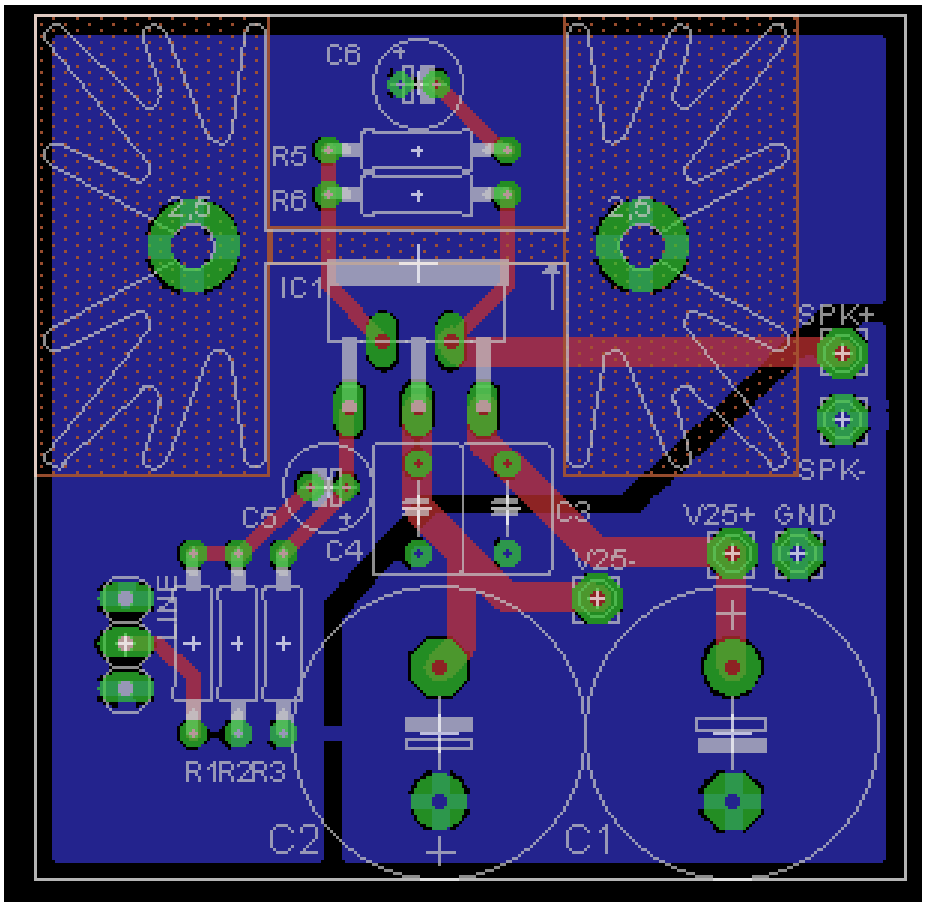

Figure 7: PCB board design of power amplifier

4.3 Power supply

The schematic of the power supply is as shown in figure 8.

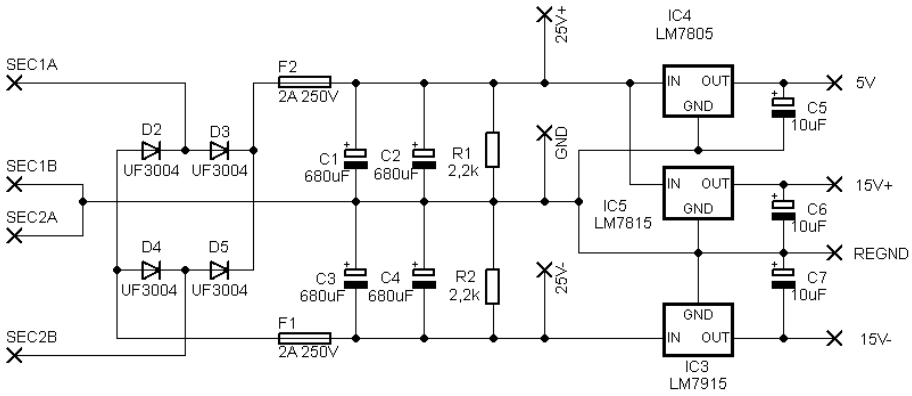

Figure 8: Schematic of power supply

The PCB design of the power supply is as shown in figure 9.

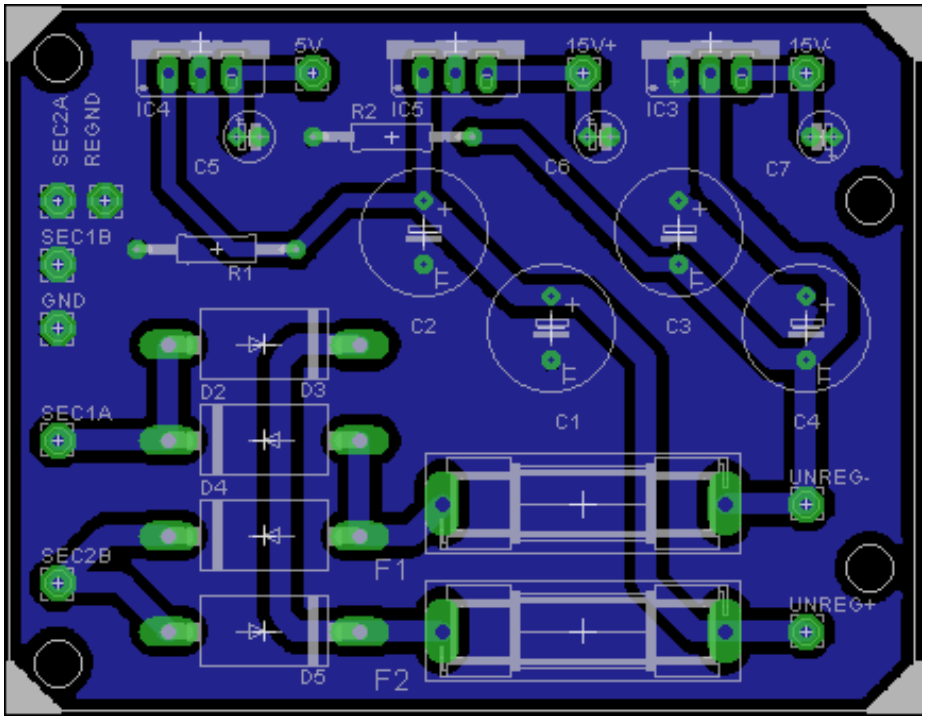

Figure 9: PCB board design of power supply

4.4 DCU

The following features and choices have been chosen/made for the implementation of the DCU:

- For the microcontroller, an ATmega328 will be used since it bears a lot of similarity with the ATmega32 which the team is most familiar with.
- The DCU will allow digital control of the balance, tone, bass, treble and volume controls through the use of digital potentiometers.
- MIDI messages will be used to pass commands to the DCU

Figure 10 shows a very basic schematic of the DCU's place within the speaker system.

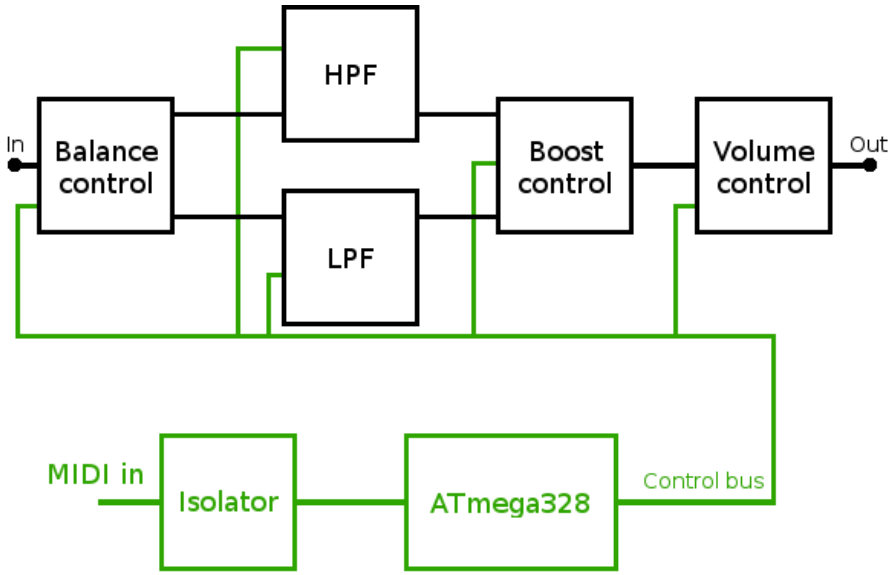

Figure 10: Block diagram of DCU

5 Preamplifier analysis

The in-depth analysis of the preamplifier consists of calculations, simulations and measurements. The calculations of the balance control, tone control and the volume control were already calculated in the measurement reports but the calculations will again be discussed in this chapter.

5.1 Calculations

5.1.1 Balance control

In this section, the calculations of the components of the balance control are presented.

Figure11 shows the position of R_H and R_L.

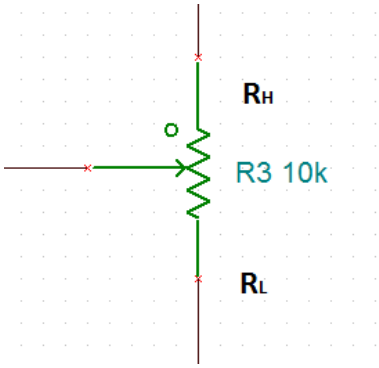

Figure 11: Positions of R_H and R_L of the potentiometer in the balance control

The formula for the gain of the balance control is equal to:

$$Gain = \frac{R_1//R_H + R_L + R_2}{R_L + R_2} = 1 + \frac{R_1//R_H}{R_L + R_2} \tag{1}$$

In the manual it is stated that the gain must be +2 dB in flat position. Therefore the gain can be derived:

$$20 \cdot log(A_{flat-position}) = 2\ dB$$

$$log(A_{flat-position}) = \frac{1}{10}$$

$$A_{flat-position} = 10^{\frac{1}{10}} \approx 1,26 \tag{2}$$

In the manual it is stated that the gain must be +6 dB if the balance control is in maximum position. Therefore the gain can be derived with the potentiometer in maximum position:

$$20 \cdot log(A_{high}) = 6\ dB$$

$$log(A_{high}) = \frac{3}{10}$$

$$A_{high} = 10^{\frac{3}{10}} \approx 2 \tag{3}$$

equation (2) can be substituted in equation (1) and R_H and R_L are 5,000 Ω.

$$1,26 = \frac{5,000 \cdot R_1}{(R_1 + 5,000) \cdot (R_2 + 5,000)}$$

$$R_2 + 5000 = \frac{5,000 \cdot R_1}{1.26 \cdot (R_1 + 5,000)}$$

$$R_2 = \frac{3,700 \cdot R_1 - 6.5 \cdot 10^6}{1.26 \cdot R_1 + 1,300)} \tag{4}$$

equation (3) can be substituted in equation (1), R_H = 10,000 Ω and R_L = 0 Ω. The value of $R_L$is actually very small (±0.5 Ω) but because the value is very small, R_L can be omitted.

$$2 = 1 + \frac{10,000 \cdot R_1}{(R_1 + 10,000) \cdot R_2}$$

$$R_2 = \frac{10,000 \cdot R_1}{(R_1 + 10,000)} \tag{5}$$

equation (5) can be substituted in equation (4) and therefore the value of R_1 can be calculated.

$$\frac{3,700 \cdot R_1 - 6.5 \cdot 10^6}{1.26 \cdot R_1 + 1,300)} = \frac{10,000 \cdot R_1}{(R_1 + 10,000)}$$

$$2,600R_1^2 + 13 \cdot 10^6 \cdot R_1 = 3,700 \cdot R_1^2 + 30.5 \cdot 10^6 \cdot R_1 - 6.5 \cdot 10^{10}$$

$$1,100R_1^2 + 17.5 \cdot 10^6 \cdot R_1 - 6.5 \cdot 10^{10} = 0$$

$$R_1 R_1 = \frac{17.5 \cdot 10^6 \pm 24,336,187}{2,200}$$

$$R_1 = 3,107.35 \approx 3,107\ \Omega \tag{6}$$

$$R_1 = -19,016\ \Omega \tag{7}$$

A resistor can never have a negative value so therefore R_1 is equal to 3,107 Ω. To calculate R_2 , equation (6) can be filled in in equation (4) or equation (5).

$$R_2 = \frac{10,000 \cdot 3,107}{(3,107 + 10,000)} = 2,370\ \Omega \tag{8}$$

5.1.2 Tone control

The tone control is divided into five sections. The tone control contains two low-pass filters, two high-pass filters and a band-pass filters as seen in figure 12.

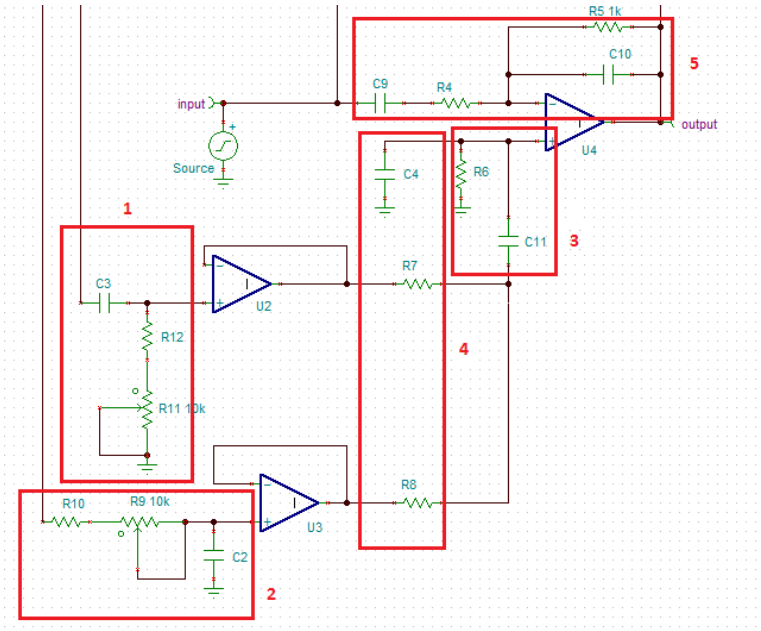

Figure 12: Tone control divided in sections

Section 1: A high-pass filter with a cut-off frequency adjustable between 2,5 kHz and 10 kHz

Section 2: A low-pass filter with a cut-off frequency adjustable between 100 Hz and 400 Hz

Section 3: A high-pass filter with a cut-off frequency of 10 Hz

Section 4: A low-pass filter with a cut-off frequency of 100 kHz

Section 5: A band-pass filter with a cut-off frequency of 2 Hz and 400 kHz

Section 1 The HPF

At a frequency of 10 kHz

$$10000 = \frac{1}{2\pi \cdot R_{12} \cdot C_3}$$

$$C_3 = \frac{1}{2\pi \cdot R_{12} \cdot 10000} \tag{9}$$

At a frequency of 2.5 kHz

$$2500 = \frac{1}{2\pi \cdot (R_{12} + 10000) \cdot C_3}$$

$$C_3 = \frac{1}{2\pi \cdot (R_{12} + 10000) \cdot 2500} \tag{10}$$

Substitute equation (9) in equations (**??**) and (**??**) to calculate resistor

$$\frac{1}{2\pi \cdot R_{12} \cdot 10000} = \frac{1}{2\pi \cdot (R_{12} + 10000) \cdot C}$$

$$R_{12} = 3333.33\ \Omega \tag{11}$$

Substitute equation (11) in equations (**??**) and (**??**) or equation (9) to calculate the capacitor

$$C_3 = \frac{1}{2\pi \cdot (3333.33 + 10000) \cdot 2500} = 4,8\ nF \tag{12}$$

Section 2 The LPF

At a frequency of 400 Hz

$$400 = \frac{1}{2\pi \cdot R_{10} \cdot C_2}$$

$$C_2 = \frac{1}{2\pi \cdot R_{10} \cdot 400} \tag{13}$$

At a frequency of 100 Hz

$$100 = \frac{1}{2\pi \cdot (R_{10} + 10000) \cdot C_2}$$

$$C_2 = \frac{1}{2\pi \cdot (R_{10} + 10000) \cdot 100} \tag{14}$$

Substitute equation (13) in equation (14) to calculate resistor

$$\frac{1}{2\pi \cdot R_{10} \cdot 400} = \frac{1}{2\pi \cdot (R_{10} + 10000) \cdot 100}$$

$$R_{10} = 3333,33\ \Omega \tag{15}$$

Substitute equation (15) in equation (13) or equation (14) to calculate the capacitor

$$C_2 = \frac{1}{2\pi \cdot (3333,33 + 10000) \cdot 2500} = 119\ nF \tag{16}$$

Section 3 The HPF

The maximum value of a ceramic capacitor available in the lab at the Hague University is 820 nF.

Capacitor C_{11} is chosen for 820 nF and with a cut-off frequency of 10 Hz for the HPF, the resistor R_6 can be calculated with the formula:

$$R_6 = \frac{1}{2\pi \cdot C_{11} \cdot 10}$$

$$R_6 = 19,409\ \Omega \tag{17}$$

Section 4 The LPF

The maximum boost or attenuation of the bass must be 10 dB at 400 Hz and the maximum boost or attenuation of the treble must be 10 dB at 2.5 kHz.

The gain of the boost is equal to:

$$Gain_{Boost} = \frac{R_7//R_8 + R_6}{R_7//R_8}$$

$$20 \cdot log(\frac{R_7|//R + R_6}{R_7|//R}) = 10\ dB$$

$$\frac{R_7//R_8 + R_6}{R_7//R} = 10^{\frac{1}{2}} \tag{18}$$

The gain of the attenuation is equal to:

$$Gain_{Attenuation} = \frac{R_7//R}{R_7//R_8 + R_6}$$

$$20 \cdot log(\frac{R_7//R_8}{R_7//R_8 + R_6}) = -10\ dB$$

$$\frac{R_7//R_8}{R_7//R_8 + R_6} = 10^{-\frac{1}{2}} \tag{19}$$

The resistors R_6, R_7 and R_8 can be any values, as long as the values of the resistors meet the requirement for the boost to be 10dB with the potentiometer in maximum position.

It is best if R_7 is equal to R_8 because the boost and attenuation of the bass and treble will differ. R_7 and R_8can be calculated by substituting the value of R_6 in equation (18) or equation (19)

$$\frac{R_7//R_8 + 19,409}{R_7//R_8} = 10^{\frac{1}{2}}$$

$$R_7//R_8 + 19409 = 3,16 \cdot R_7//R_8$$

$$R_7//R_8 = 8,900\ \Omega$$

$$R_7 = R_8 = 17,800\ \Omega \tag{20}$$

With a cut-off frequency of 100 kHz for the LPF in section 4, the capacitor C_4 can be calculated with the formula:

$$C_4 = \frac{1}{2\pi \cdot (R_7//R_8) \cdot 100 \cdot 10^3}$$

$$C_4 = 180\ pF \tag{21}$$

Section 5 The BPF

The frequency range must be between 2 Hz and 400 kHz and the gain at these frequencies must be -3 dB. Therefore the resistors and capacitors can be calculated for the band-pass filter with the formula:

$$f_{cut-off} = \frac{1}{2\pi \cdot R \cdot C} \tag{22}$$

For the design of the band-pass filter, a value of 820 nF is chosen for capacitor C_9. The resistors can now be calculated.

For the HPF part of the band-pass filter is the resistor R_4 equal to:

$$R_4 = \frac{1}{2\pi \cdot 2 \cdot 820 \cdot 10^{-9}} = 97,045\ \Omega \tag{23}$$

The balance control and the volume control regulate the gain of the preamplifier. Therefore the band-pass filter does not need to amplify the signal. Thus it is neccessary for R_4 to be equal to $R_5$to attain a gain of 1 (0 dB).

For the LPF part of the band-pass filter, the capacitor is equal to:

$$C_{10} = \frac{1}{2\pi \cdot 400 \cdot 10^3 \cdot 97,045} = 4.1\ pF \tag{24}$$

5.1.3 Volume control

With an input voltage of 200mV, an output voltage of 2V must be acquired with the balance control in flat position, tone (bass and treble) control switched off and the volume control at maximum. With a gain of approximately 1.26 in the balance control and a gain of 1 in the tone control, the voltage at the input of the volume control is approximately equal to:

$$U_{input-volume-control} = 1.26 \cdot 1 \cdot 200mV = 252\ mV \tag{25}$$

The input voltage of the volume control is now known and because the volume control is at maximum, the voltage at the output of U_5 is equal to the input voltage of the volume control. The values of R_{17} and R_{18} can be calculated.

$$\frac{R_{18}}{R_{17}} \cdot U_{input-volume-control} = 2\ V$$

$$\frac{R_{18}}{R_{17}} \cdot 0.252V = 2\ V$$

$$\frac{R_{18}}{R_{17}} = 7.94 \tag{26}$$

The ratio between R_{18} and R_{17} is 7.94. The values of R_{18} and R_{17} can be determined. The following values have been chosen for R_{18} and R_{17}:

$$R_{18} = 7,940\ \Omega \tag{27}$$

$$R_{17} = 1,000\ \Omega \tag{28}$$

With the volume control at minimum, the gain should be very low (-∞ dB) but that is impossible in practice. Therefore simulations are required to determine what the gain can be if the volume control is almost set to the minimum (99%). The gain with the volume control set almost to the minimum is approximately -50 dB. The gain with the potentiometer in middle position (50%) is equal to:

$$17.95dB - (\frac{17.95dB - -50dB}{2}) = -16\ dB$$

Resistor R_{16} can be calculated to acquire an approximate logarithmic control function:

$$20 \cdot log|\frac{R_{16}}{5,000 + R_{16}} \cdot \frac{7,940}{1,000}| = -16\ dB$$

$$log(\frac{R_{16}}{5,000 + R_{16}} \cdot \frac{7,940}{1,000}) = -\frac{16}{20}$$

$$\frac{R_{16}}{5,000 + R_{16}} \cdot \frac{7,940}{1,000} = 10^{-\frac{16}{20}}$$

$$R_{16} = 100\ \Omega \tag{29}$$

5.2 Simulations

Figure 13 shows a simulation of the bode plot of the preamplifier with the bass and treble both at maximum boost, the volume control in mid position, the balance control in mid position, the bass corner frequency at 400 Hz and the treble corner frequency at 2.5 kHz.

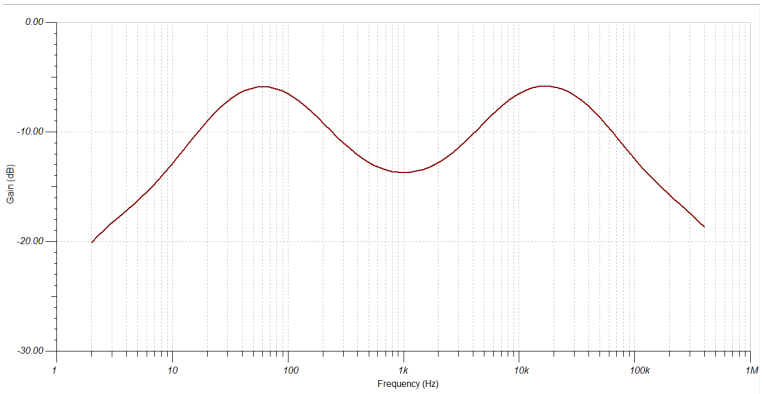

Figure 13: Bode plot of the preamplifier in maximum boost

Figure 14 shows the simulation of the bode plot of the preamplifier with the bass and treble both at maximum attenuation, the volume control in mid position, the balance control in mid position, the bass corner frequency at 400 Hz and the treble corner frequency at 2.5 kHz.

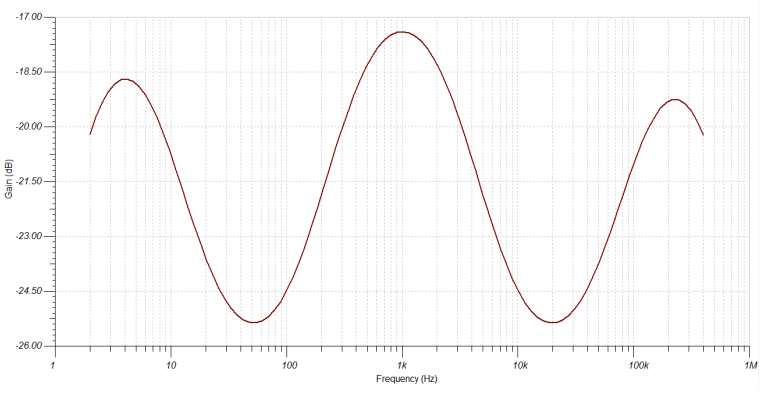

Figure 14: Bode plot of the preamplifier in maximum attenuation

5.3 Measurements

Figure 15 shows a bode plot of the preamplifier with the bass and treble both at maximum boost, the volume control in mid position, the balance control in mid position, the bass corner frequency at 400 Hz and the treble corner frequency at 2.5 kHz. The measurements for determining the bode plot in figure 15 are in table 4 in Appendix A.

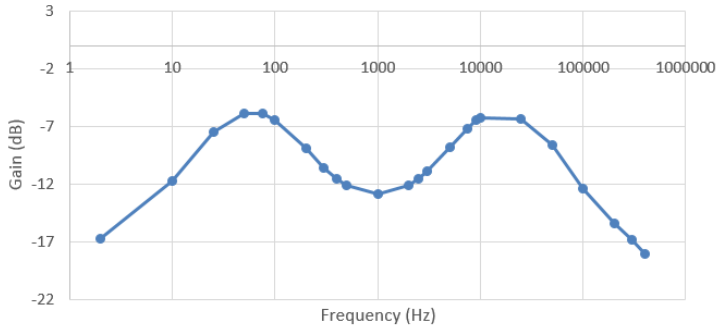

Figure 15: Bode plot of the preamplifier in maximum boost

Figure 16 shows a bode plot of the preamplifier with the bass and treble both at maximum attenuation, the volume control in mid position, the balance control in mid position, the bass corner frequency at 400 Hz and the treble corner frequency at 2.5 kHz. The measurements for determining the bode plot in figure 16 are in table 3 in Appendix A.

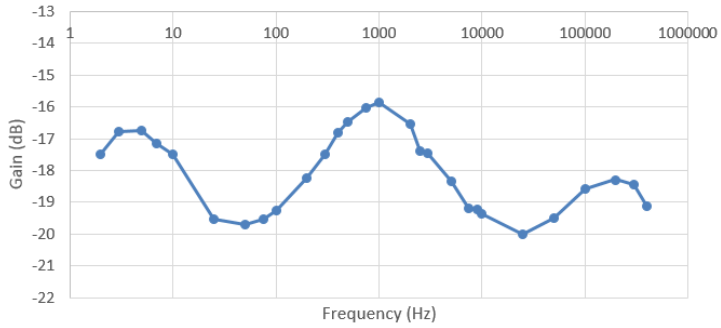

Figure 16: Bode plot of the preamplifier in maximum attenuation

The bode plots of the measurements correspond with the bode plots of the simulations.

6 Power supply design

The power supply takes 230V mains and converts it to lower voltages to power the various modules in the speaker system. First a quick design overview is given and the schematic is explained. Then the individual stages of the supply are discussed and finally the energy efficiency and board layout are explained.

6.1 Design overview

The power supply is a linear non-switching supply, which uses a toroidal transformer to take 230V mains down to 36V. The power amplifier and the preamplifier require a split supply by design, so a transformer with two secondary windings is used. After the transformer comes a bridge rectifier which converts the AC signal to DC. Then finally the signal will pass through several filter capacitors to smooth out the voltage.

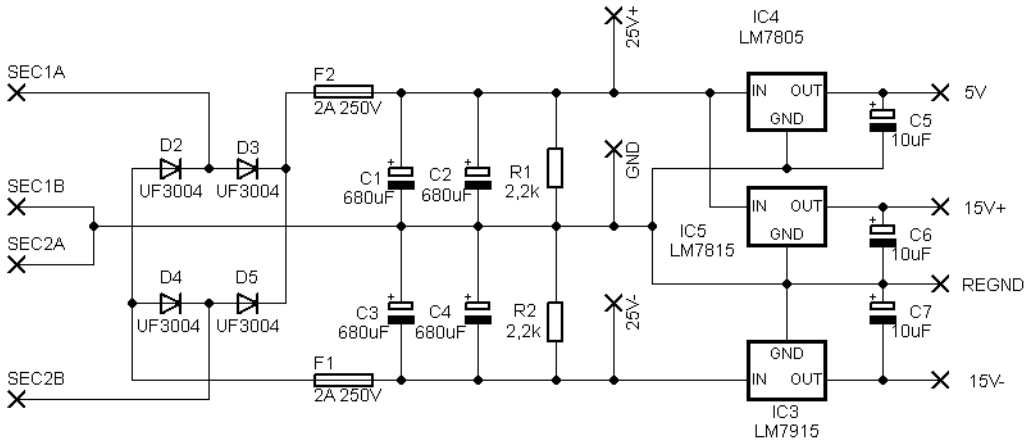

Figure 17: Power supply schematic

6.2 Filter section

Per rail two filter capacitors are used in order to reduce the ripple voltage caused by the equivalent series resistance. A bypass resistor is put in parallel with the capacitors, so they discharge safely when the power supply is switched off even with no load on the output.

The nominal voltage can be calculated as follows:

$V_{in,pk-pk} = V_{in} * \sqrt{2} = 36 * \sqrt{2} = 51V$

$Vout = V_{in,pk-pk} - 2V_{d(on)} = 51 - 1.4 = 50V$

At full power output the ripple voltage on the positive and negative supply can be calculated:

$T_{on} = \frac{1}{2f} = \frac{1}{100}$

$\Delta Vout = \frac{I*T_{on}}{C} = \frac{0.7*0.01}{2*680*10^{-6}} = 5V$

According to the power amplifier's specifications this is an acceptable ripple voltage. Additional filter capacitors were added close to the power amplifier to reduce the loop size and increase stability. With these additional capacitors in place the ripple voltage becomes:

$\Delta Vout = \frac{I*T_{on}}{C} = \frac{0.7*0.01}{(2*680+2200)*10^{-6}} = 2V$

6.3 Protection

The power supply is fused on both secondary windings to protect the transformer against high currents. The nominal current of the secondary windings of the transformer is 0.420A while the power amplifier takes 0.694A from the supply at max power . The capacitors in the power supply will smooth out the load on the transformer, so it does not exceed the nominal current. The fuses on the secondary windings are rated 2A, so they only blow on a short circuit. The primary winding could be fused optionally.

6.4 Regulator section

The preamplifier and DCU require lower voltages, so linear voltage regulators are used to step down the voltage. For the power amplifier, however an unregulated supply is used. The regulators require small capacitors between the output terminals to stabilize the output voltage, so a few 10uF capacitors were added.

6.5 Efficiency

As said before, the power amplifier's power rails are unregulated. This is required to meet the 80% energy efficiency requirement. If linear regulators were used to regulate 25 down to 20V at 0.5A this would result in a 2.5W loss in the

regulator. In this case the energy efficiency would be 80% for the regulator stage alone. The power dissipated in the linear regulators for the preamp and dcu is neglectable, because these parts take very little current compared to the power amplifier. The overall efficiency of the power supply is equal to the efficiency of the transformer multiplied by that of the diode bridge rectifier.

$\eta_{diode} = \frac{V_{in,pk-pk} - 2V_{d(on)}}{V_{in,pk-pk}} = \frac{51-1.4}{51} = 0.97$

$\eta_{transformer} = 0.87$

$\eta_{total} = \eta_{diode} * n_{transformer} = 0.85$

Therefore the overall efficiency is 85%.

6.6 Board layout

The power supply can be constructed on a single sided board to reduce manufacturing cost. The components are all through-hole, because small surface mount parts cannot handle the high voltages and currents. It also makes soldering a prototype by hand easier.

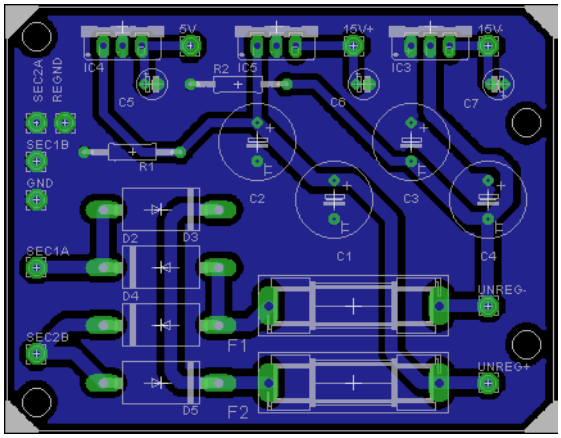

Figure 18: Power supply board layout

7 DCU design

The DCU's entire functionality was to be determined by the project team. The only requirements given by BARK were the use of an ATmega32 microcontroller or similar controllers, and some kind of interaction with the analogue part of the system. In team 1's case, the DCU allows control of all audio controls (potentiometers) in the analogue circuits using MIDI messages. The DCU then adjusts the values of digital potentiometers placed in the circuits based on the incoming MIDI message. First, the physical design of the DCU and related components are discussed. After that, the code used for the DCU will be looked at.

7.1 The DCU, related components, and its connection to the rest of the system

The choice to use MIDI messages as input for the DCU was made with the idea that MIDI hardware is generally designed for use with music and/or audio. On top of that, professional mixing devices can be controlled using computer software, allowing preset settings to be loaded in a matter of a few clicks. Many MIDI devices include twisting knobs which act as potentiometers, and thus being able to use a MIDI device to control the various controls in the circuitry allows for an intuitive and easily expandable/customizable user interface. The type of digital potentiometers used in combination with the DCU are the MCP4251-103E/P (see figure 19).

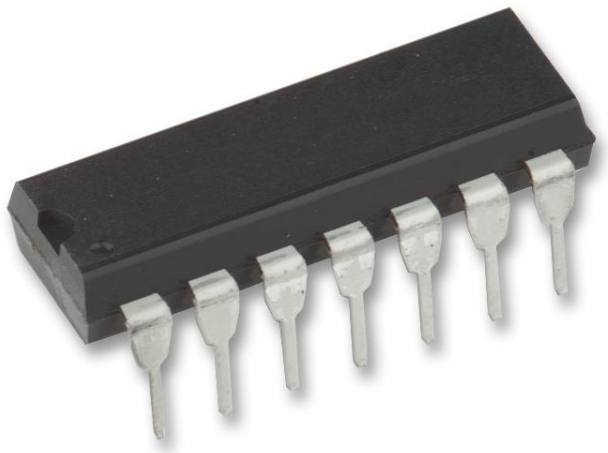

Figure 19: The MCP4251-103E/P

These models hold two potentiometers per package, and expect incoming commands through SPI which makes them easily usable with the ATmega328 which supports a direct SPI connection. As illustrated in figure 20, the digital potentiometers allow the DCU to take over control of the balance control, bass cut-off control, treble cut-off control, bass boost control, treble boost control and volume control. In its current state, the code only handles so-called 'Control Change' MIDI messages which indicate the change of a physical control, such as a knob. These messages include a value indicating the position of the knob as well as an identifier indicating which specific knob was changed. The message is processed in the DCU and the correct command is passed on to the appropriate potentiometer over the SPI connection. Only when a new 'Control Change' message is received, will the digital potentiometer(s) be adjusted accordingly.

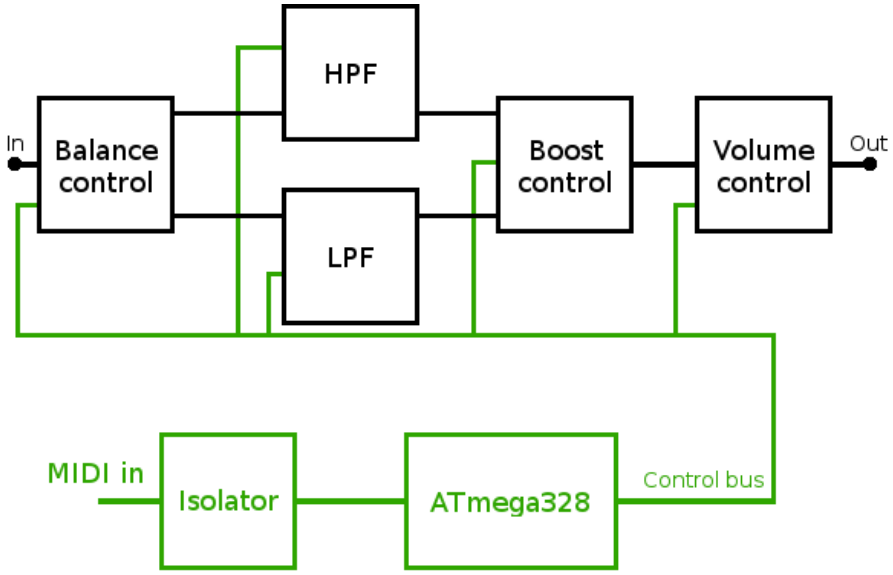

Figure 20: Block diagram of DCU

As seen in figure 20, the control bus (ATmega328 SPI output) is connected to all parts of the preamplifier. The control bus consists of seven signal lines: six CS (chip select) lines and one data line (carrying the value to be read by the digital potentiometer. The MIDI input and the USART of the ATmega328 are electrically isolated using an optocoupler. This protects both devices from overvoltage and static discharges. It also makes it possible to swap out midi cables while the speaker system is powered on. Figure 21 shows the schematic of the DCU part.

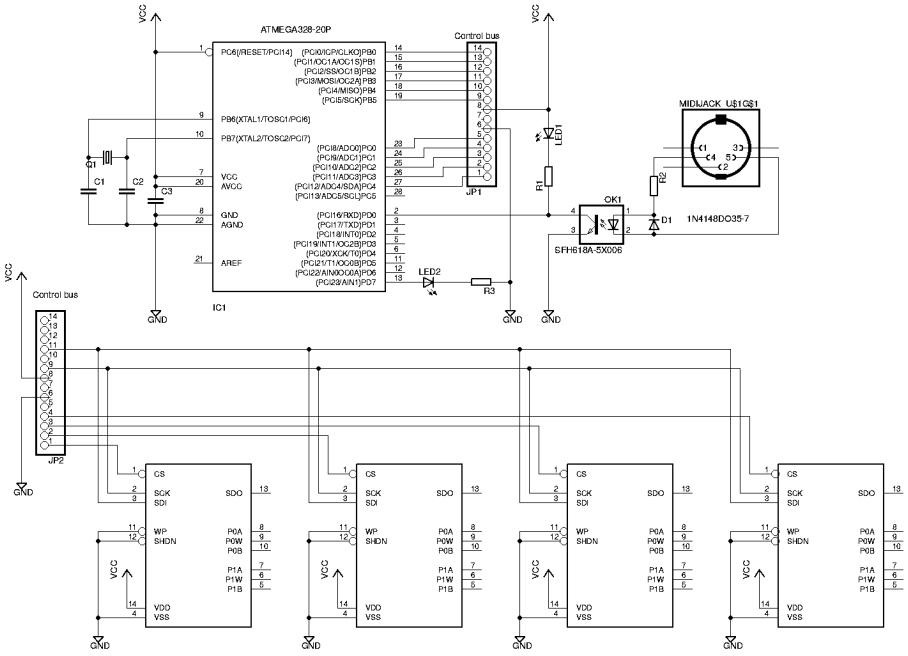

Figure 21: Schematic of DCU

7.2 C code

The C code can be divided into the following subsections:

- Headers for digital potentiometers and MIDI
- Initialization
- Main program loop
- MIDI input functions
- Digital potentiometer output functions

The individual blocks of code will be discussed and explained. In order to see how the different parts of the code are connected to each other, the complete C code can be found in Appendix B. When addressing specific pieces of code, references will be made to the line numbers as seen in these appendices.

7.2.1 digipots.h and midi.h

The two custom header files used in the code contain all the constants asssociated with the ports (and pins) connected to the peripheral devices. As seen in Appendix B, digipots.h contains the correct values to set the DDR and PORT registers associated with the potentiometers. The `*_CS_PIN` constants define the correct value to be written to the PORT register that drive the $\overline{CS}$ pins of the digital potentiometers. Since there is only one data line for all of the controls, when writing a value to the digital potentiometers, only the appropriate $\overline{CS}$ pins are to be driven. This will become more clear when the output functions are discussed. midi.h contains all the constants associated with the incoming MIDI data. MIDI messages consist of at least one byte. In case of MIDI Control Change messages which we are concerned with, a full message consist of three bytes: a command byte `1011nnnn`, and Controller Number byte `0ccccccc` and a Controller Value byte `0vvvvvvv` respectively where `nnnn` denotes the MIDI channel, `ccccccc` denotes the controller number and `vvvvvvv` denotes the value of the controller. `CTRL_CH` is the byte value in the MIDI message that denotes the type of MIDI command (in our case, Control Change). The `*_CTRL` constants are used to define what controller number (in the `0ccccccc` byte) correspond to what control. This way the program will be able to determine which $\overline{CS}$ pin should be driven when the received controller value is passsed on to the digital potentiometers.

7.2.2 Initialization

The initializations are made in dcu.c, from the beginning to line number 66. The global variables are declared, appropriate DDR registers are set, the USART registers are set, and the SPI registers are set. Finally, all controls are reset by initializing the digital potentiometers to maximum position. The appropriate values to be loaded into the UBRRH and UBRRL registers of the USART have been calculated based on the values defined in line numbers 20 to 24. The for-loop on lines 61 to 63 set the digital potentiometers to the maximum value one by one.

7.2.3 Main program loop

The main program loop, an infinitely looping while-loop from lines 69 to 106, calls the appropriate functions to control the digital potentiometers whenever new data is received from the MIDI device through the USART. The program will wait at the `while (rxcnt == 0);` lines for a new byte to arrive through the USART. On line 73, a command byte will be expected. If a new byte is received, it is stored into `cmd`. On line 74, the command byte is checked for validity.

If valid, the program continues and the error LED is turned off. Otherwise, the program will end up waiting at line 72 again for a valid command byte. After passing the command validity check, the same process is repeated for the Controller Number and Contorller Value bytes. If either of these checks fail, the else-statement is executed and a LED is switched on using `toggleErrorLED()`. When all received bytes are valid (making up a complete valid three-byte Control Change message), a switch case at lines 83 to 95 will select the appropriate action. There is only one case in this case (excluding the default case), but the system can be expanded in the future to include more functionality using this switch case.

7.2.4 MIDI input functions

There are two blocks of code dealing with the incoming MIDI messages. They are the interrupt service routine (ISR) `USART_RXC_vect` (lines 151 to 169) and the function `ugetc()` (lines 185 to 192). Whenever a MIDI device sends a byte to the ATmega328, the ISR is executed and the incoming data is checked. If the USART deems the data valid, the UDR is unloaded into variable `c` and stored into the buffer `sb` if it is not full yet. Finally, `rxcnt` is incremented. One can now see that in the main loop, the program will only pass the 'waiting' lines when `rxcnt` is not equal to zero, which means one or more unread bytes are stored in the buffer.

When a message byte needs to be stored into the variables in the main loop, `ugetc()` is called. This function selects the correct byte in the buffer and stores it in local variable `c`. It then decrements `rxcnt` and returns `c`.

7.2.5 Digital potentiometer output functions

There are two functions related to the output controlling the digital potentiometers. They are `spiSend()` (lines 112 to 117) and `setPot()` (lines 119 to 139). When `setPot()` is called, a switch case chooses the appropriate case based on the parameter `potno`. Inside each case, the appropriate $\overline{CS}$ pin is driven and `spiSend()` is called with the appropriate parameters.

`spiSend()` takes two arguments, the memory address value `memcom` and the control value `val`. This function does nothing more but transmitting the two bytes over SPI one by one. The while statements force the program to wait with continuing untill the SPI is finished with he transmission.

8 Test procedures and test results

Figure 22 shows the complete loudspeaker system. This active loudspeaker system was tested using the procedures below.

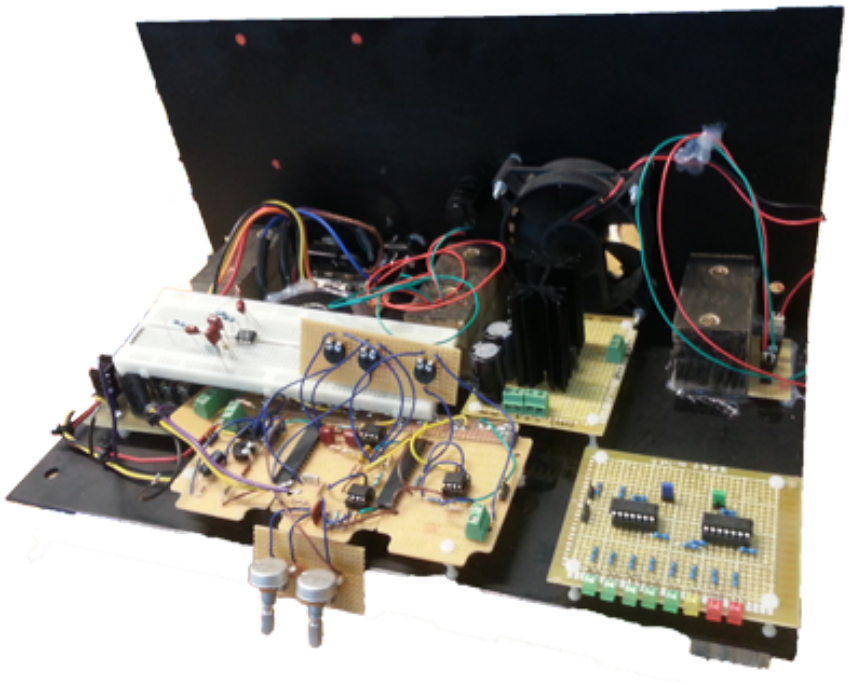

Figure 22: The built active loudspeaker system

8.1 Equipment

The following equipment was used to test the active loudspeaker system:

- Audio source (e.g. phone)

- Audio cable
- Power cable
- 'TEKTRONIX DPO2004B' oscilloscope
- 8Ohm speaker

8.2 Setup

- Remove all power and audio cables from the speaker system and make sure the power switch is turned off before proceeding.
- Connect the speaker to the output of the power amplifier.
- Connect the audio source to the input of of the preamplifier.
- Connect the power cable to the input of the power supply.
- Make sure that everything is connected before switching the power on.
- Switch the power on.

8.3 Results

Table 1: The result of the test

#	Procedure	Expected Results	Pass / Fail / Not Te
	Play music from the audio source	Hear music coming from the speaker regardless the performance of the sound.	Pass
1	Test the volume of the preamplifier.	No audible sound when the volume control adjust is turned down to the minimum. The sound will get louder as the volume control is turned up to the maximum	Pass
2	Is the sound clear?	No distortion, no clipping, little mains hum	Pass
3	Test the bass of the preamplifier	Low frequencies will be heard when the bass is in maximum boost.	Pass
4	Test the treble of the preamplifier	High frequencies will be heard when the treble is in maximum boost	Pass
	Switch off the unit and remove the heatsink.		-
5	Turn up the volume. Does the thermal shutdown feature of the amplifier work?	The amplifier should shut down when it reaches 170°C	Pass
			-
6	Test the DCU system		Not Tested

8.4 Notes on the test

There were some problems with test number 2 in table 1. The sound coming from the speaker temporarily shuts down (approximately two seconds) and turns back on. This problem happens in an interval between 10 to 15 seconds. This problem is solved by measuring the outputs of the power amplifier and the preamplifier and comparing the signals during the shutdown of the sound. During shutdown there is no signal at the power amplifier but there is a signal at the preamplifier. This means that the problem lies within the power amplifier. After examining the power amplifier, it was discovered that the transistor wasn't attached well to the heatsink, so it would become too hot and shut itself down. After tightening the bolt that clamps the heatsink to the power amplifier there have been no more thermal shutdowns.

9 Total costs of system

The following components are required to build the power amplifier, preamplifier, power supply and digital control unit. Please note that these prices are only an indication. Prices may change over time.

Name	Amt	Farnell No,	Price Each	Price
Power supply				
Power switch	1	1839427	4,45	4,45
Transformer	1	9530380	24,98	24,98
D2/D3/D4/D5:	4	1861507	0,0638	0,2552
Power connector	1	2080459	0,95	0,95
Fuse holder	3	1162740	0,563	1,689
C1/C2/C3/C4 : 680u	4	1800669	0,26	1,04
C5/C6/C7: 10u	3		0,1	0,3
R1/R2: 2.2k	2		0,02	0,04
IC4:LM7805CT	1	1102157	0,84	0,84
IC5:LM7815CT	1	2296025	0,71	0,71
IC3:LM7915CT	1	2296026	1,22	1,22
HEATSINK	3	1710627	0,506	1,518
Fuse 2A,250V	3	1123175	0,183	0,549
Power amplifier				
LM1875T	1	1468913	3,36	3,36
HEATSINK	1	1319813	5,29	5,29
R1: 100	1		0,02	0,02
R2: 1M	1		0,02	0,02
R3: 56k	1		0,02	0,02
R5: 1k	1		0,02	0,02
R6: 22k	1		0,02	0,02
C1/C2: 2200uF	2	2346523	1,63	3,26
C3/C4: 100nF	2		0,1	0,2
C5: 2.2uF	1		0,1	0,1
C6: 22uF	1		0,1	0,1
Preamplifier				
Phono Jack	1	148266	2,38	2,38
NE5532P	4	1106092	0,79	3,16
R1: 3.1k	1		0,02	0,02
R2: 2.4k	1		0,02	0,02
R4/R5: 97k	2		0,02	0,04
R6: 19k	1		0,02	0,02

R7/R8: 18k	2		0,02	0,04
R10/R12: 3.33k	2		0,02	0,04
R16: 100	1		0,02	0,02
R17: 1k	1		0,02	0,02
R18: 7.9k	1		0,02	0,02
C2: 120n	1		0,1	0,1
C3: 4.8n	1		0,1	0,1
C4: 180p	1		0,1	0,1
C5/C6/C7/C8: 22u	4		0,1	0,4
C9/C11: 820n	2		0,1	0,2
C10: 4p	1		0,1	0,1
C12/C13: 220u	2		0,1	0,2
C14...C22: 100nF	8		0,1	0,8
IC socket DIP8	4	2445620	0,195	0,78
Digital control unit				
IC1: ATMEGA328-PU	1	1972087	2,36	2,36
MCP4251-103E/P	4	1578442	0,921	3,684
IC socket DIP28	1	2445626	0,44	0,44
IC socket DIP14	4	2445621	0,198	0,792
Crystal 16MHz	1	1611761	0,378	0,378
C1/C2: 22pF	2		0,1	0,2
LED1/LED2: red led	2	2112100	0,214	0,428
D1:1N4148	1	9565124	0,0203	0,0203
R1/R2/R3: 150	3		0,02	0,06
VO617A-2	1	2251550	0,187	0,187
DIN5 jack	1	1280804	1,45	1,45
C3: 100n	1		0,1	0,1
IC socket DIP6	1	2445619	0,204	0,204
Misc				
TERMINAL BLOCK	10	3041440	0,819	8,19
Pin headers m20	2	671990	1,6	3,2
Pin headers f20	1	1577768	2	2
PCB single sided	1	149056	13,46	13,46
Misc wire ass.	1		2	2
		Total (ex vat):		?98, 66
		Total:		?119, 38

10 Conclusions and recommendations

After approximately 2 months of working on PRO-Q2, the active loudspeaker was finally finished by connecting the power supply, the power amplifier and the preamplifier. After testing the active loudspeaker as a whole, the team concluded that the active loudspeaker works. The sound may not be perfect but at the very least the speaker is able to reproduce sound without excessive distortions. Unfortunately the DCU was not tested.

The bass and treble also works well when turning the bass and treble on and off. The volume control works well when turning the volume off and on. The balance control works when turning the balance control on and off. The preamplifier was simulated and measured and the bode plots of the simulations and measurements correspond with each other as seen in section 5. The conclusion can be made that the calculations of the components of the preamplifier are correct and that the preamplifier meets the specifications.

The power supply meets all specifications and it supplies all voltages required to make the system work.

The power amplifier meets all specifications, its temperature stays within the accepted range and it produces clean audio.

As the source code in the appendices already notes, at the time of writing the DCU had not been connected and tested yet. Building the code resulted in the expected errors associated with the yet-to-be-defined values in the header files, but all other errors that were previously present have been succesfully ironed out. In short, source-code-wise the DCU is fully finished, although its functionality is yet to be proven.

References

[1] J. op den Brouw, M. Can. (2015, May). "*Project Manual PRO-Q2 (version 1.3)*" [Online]. Available: https://blackboard.hhs.nl/bbcswebdav/pid-1650195-dt-content-rid-3258337_2/xid-3258337_2 [June 12, 2015]

[2] Multicomp. (2014, June). "*Toroidal Transformers, General Purpose*" [online]. Available: http://www.farnell.com/datasheets/182927 8.pdf [June 15, 2015]

Appendix A

Table 3: Measurements of the preamplifier maximum attenuation

f (Hz)	V_{in} (mV)	V_{out} (mV)	Gain (dB)
2	90	12	-17.50122527
3	89.8	13	-16.78665969
5	90	13.1	-16.73942428
7	90	12.5	-17.14664993
10	90	12	-17.50122527
25	90	9.5	-19.53037808
50	90	9.3	-19.71519122
75	90	9.5	-19.53037808
100	90	9.8	-19.26032867
200	90	11	-18.25699649
300	90	12	-17.50122527
400	90	13	-16.80598314
500	90	13.5	-16.47817482
750	90	14.2	-16.0390833
1000	90	14.5	-15.85749014
2000	90	13.4	-16.54275422
2500	91	12.3	-17.38272562
3000	91	12.2	-17.45363123
5000	91	11	-18.35297414
7500	91	10	-19.18082785
9000	90.4	9.9	-19.21066472
10000	91	9.8	-19.35630633
25000	92	9.2	-20
50000	91.5	9.7	-19.4929872
100000	91.8	10.8	-18.58837851
200000	92	11.2	-18.29139609
300000	92	11	-18.44790284
400000	92.2	10.2	-19.12261499

Table 4: Measurements of the preamplifier maximum boost

f (Hz)	V_{in} (V)	V_{out} (V)	Gain (dB)
2	90	13.1	-16.73942428
10	90	23.4	-11.70053304
25	90	38.1	-7.466350675
50	90	45.9	-5.848596478
75	90	45.9	-5.848596478
100	90	42.9	-6.435704345
200	90	32.5	-8.847182969
300	90	26.7	-10.55462496
400	90	23.8	-11.55331105
500	90	22.3	-12.11875293
1000	90	20.4	-12.89224684
2000	90	22.3	-12.11875293
2500	90.5	24	-11.52874675
3000	90.6	25.9	-10.87656867
5000	91	33	-8.810549049
7500	91	39.9	-7.161369933
9000	90	43.1	-6.395304786
10000	91	44.2	-6.272382459
25000	92	44.3	-6.347682022
50000	91.7	33.9	-8.643392749
100000	92	22	-12.42730293
200000	92	15.6	-15.41326458
300000	92	13.2	-16.86427792
400000	92.2	11.5	-18.08066161

Appendix B

The following pages show all C code involved used in the ATmega328 microcontroller to make the DCU function. The first two pages show the two .h files where several constants such as ports, pins, and MIDI-related values. Then, the main program file follows showing the main program loop and all used functions.

```
1  /* TO-DO: fill in all correct values for everything. Also consider whether PORTs are
   active high/low. */
2  #ifndef DIGIPOTS_H
3      #define DDRDIGIPOTS         DDRx    /* Data Direction Register for DigiPots */
4      #define PORTDIGIPOTS        PORTx   /* Port for DigiPots */
5
6      #define DDRLEDS             DDRx    /* Data Direction Register for LED(s) */
7      #define PORTLEDS            PORTx   /* Port for LED(s) */
8
9      #define VOLUME_CS_PIN       0x00    /* Pin for volume control CS */
10     #define BALANCE_CS_PIN      0x00    /* Pin for balance control CS */
11     #define BASS_CUT_CS_PIN     0x00    /* Pin for bass cut control CS */
12     #define TREBLE_CUT_CS_PIN   0x00    /* Pin for treble cut control CS */
13     #define BASS_BOOST_CS_PIN   0x00    /* Pin for bass boost control CS */
14     #define TREBLE_BOOST_CS_PIN 0x00    /* Pin for treble boost control CS */
15
16     #define DIGIPOTVAL_PIN      0x00    /* Pin for the value to be loaded into the
       DigiPot */
17 #endif
18
```

```
1  /* TO-DO: fill in all correct values for CTRL numbers */
2  #ifndef MIDI_H
3      #define CTRL_CH           0xB0  /* Midi controller value change */
4
5      #define BALANCE_CTRL      0x00  /* Control number of knob used for balance
       control */
6      #define BASS_CUT_CTRL     0x00  /* Control number of knob used for bass cut
       control */
7      #define TREBLE_CUT_CTRL   0x00  /* Control number of knob used for treble cut
       control */
8      #define BASS_BOOST_CTRL   0x00  /* Control number of knob used for bass boost
       control */
9      #define TREBLE_BOOST_CTRL 0x00  /* Control number of knob used for treble boost
       control */
10     #define VOLUME_CTRL       0x00  /* Control number of knob used for volume
       control */
11 #endif
12
```

```
/* Filename: dcu.c
 * Authors: Kevin Oei, Koen van Vliet
 * For Pro-Q2
 * Description: Digital control unit firmware for speaker system
 * Status: -
 * Notes:   - Check for MSB in parameters. If high: error condition
 *          - Implement status LEDs
 */
      615
 /* TO-DO: Set the DDRs correctly (optional), put the right memory address values in setPot(), use the right value in toggleErrorLED() */

#include <avr/io.h>
#include <stdio.h>
#include <avr/interrupt.h>
#include "midi.h"
#include "digipots.h"
#include "uart.c"


#define SYSFREQ 16000000
#define BAUDRATE 31250
#define RATE SYSFREQ / (2*BAUDRATE) - 1
#define RATE_L RATE%256
#define RATE_H RATE/256


char sb[64];                /* The buffer for the USART input. Incoming MIDI messages are stored in here. */
volatile char rxcnt = 0;    /* rxcnt is used when determining the offset, which is used to pick the right index when reading sb */
volatile char rxp = 0;      /* rxp will loop from (decimal) 0 to 63, used in determining the index of sb that needs to be approached */


void uputc(char c);
void uputs(char str[]);
char ugetc();
void spiSend(uint8_t memcom, uint8_t data);
void setPot(int potno, uint8_t val);
void toggleErrorLED(int onoff);


int main(void) {
    int i;

    /* Debug LEDs */
    DDRB        = 0xFF;
    DDRDIGIPOTS = 0xFF;
    DDRLEDS     = 0xFF;
    /* Setup serial comms (31250-8-n-1) NOTE: THIS IS FOR PC ONLY, HAVE TO FIX FOR ATmega */
    UCSRA = 0x00;
    UCSRB = 0x18 | (1<<7);  /* Enable receiver & transmitter and RX complete interrupt enable */
    UCSRC = 0x86;           /* STILL NEEDS TO BE ADJUSTED WHEN TO-BE-USED DEVICE IS KNOWN */
```

```
51      UBRRH = RATE_H;
52      UBRRL = RATE_L;
53
54      /* Setup SPI bus (f/2, mode0.0) */
55      SPSR = (1<<0);                      /* Double SPI speed enabled */
56      SPCR = (1<<7) | (1<<6) | (1<<4);    /* Enable SPI and interrupt, select master
        mode */
57
58      /* Reset digital potentiometers (255 is the max position) */
59      static uint8_t potpos[6] = {255};
60      /* Connect terminals */
61      for (i = 0; i < sizeof(potpos); i++) {
62          setPot(i,potpos[i]);
63      }
64
65
66      sei();
67
68      /* Main program loop */
69      while (1) {
70          uint8_t cmd, cc, vv;
71          char s[50]; /* Is still being used anywhere? */
72          while (rxcnt == 0); /* Wait for buffer to be empty */
73          cmd = (uint8_t)ugetc();
74          if (cmd & 0x80) { /* Check if 0b1xxxxxxx (valid MIDI command) */
75              toggleErrorLED(0); /* Turn off error LED */
76              while (rxcnt == 0);
77              cc = ugetc(); /* Acquire controller number */
78              if (~cc & 0x80) { /* Check if 0b0xxxxxxx (valid controller number value)
                */
79                  while (rxcnt == 0);
80                  vv = ugetc(); /* Acquire controller value */
81                  if (~vv & 0x80) { /* Check if 0b0xxxxxxx (valid controller value) */
82                      /* Check command type */
83                      switch (cmd) {
84                          case CTRL_CH:   snprintf(s, sizeof(s), "Controller %d = %d",
                          cc, vv);
85                                          uputs(s);
86                                          potpos[cc] = vv;
87                                          setPot(cc,potpos[cc]);
88                                          break;
89                          /*case 'r':     for (i = 0; i < sizeof(potpos); i++) {
90                                              potpos[i] = 127;
91                                              setPot(i,potpos[i]);
92                                          }
93                                          break; */
94                          default:        uputs("What?");
95                      }
96                  }
97                  else {
98                      toggleErrorLED(1); /* Turn on error LED */
99                  }           }
100             else {
101                 toggleErrorLED(1); /* Turn on error LED */
102             }
103         }
104     }
```

```
105         return 0;
106     }
107
108     /* The messages that are sent out are 16-bit long. SPI can only send 8-bit at one
        time */
109     /* Thus, two SPI transmissions are required for a full message to be sent */
110     /* The message is as follows: AAAA.CCDD.DDDD.DDDD where A is memory address, C is
        command and D is data. */
111     /* See pg.47 of DigiPot datasheet */
112     void spiSend(uint8_t memcom, uint8_t data) {
113         SPDR = memcom;                  /* Transmit memory address and command */
114         while ((SPSR & (1<<7)) == 0);   /* Wait for SPI transfer to finish. */
115         SPDR = data;                    /* Transmit data (the value to be written to the
            DigiPot) */
116         while ((SPSR & (1<<7)) == 0);   /* Wait for SPI transfer to finish. */
117     }
118
119     void setPot(int potno, uint8_t val) {
120         /* Select the CS of the correct IC. Note that a low signal will 'activate' the
            IC. */
121         /* ADDRESS MEMORY STILL NEEDS TO BE FILLED IN */
122         switch (potno) {
123             case 0: PORTDIGIPOTS = VOLUME_CS_PIN;
124                     spiSend(0x00, val);
125             case 1: PORTDIGIPOTS = BALANCE_CS_PIN;
126                     spiSend(0x00, val);
127             case 2: PORTDIGIPOTS = BASS_CUT_CS_PIN;
128                     spiSend(0x00, val);
129             case 3: PORTDIGIPOTS = TREBLE_CUT_CS_PIN;
130                     spiSend(0x00, val);
131             case 4: PORTDIGIPOTS = BASS_BOOST_CS_PIN;
132                     spiSend(0x00, val);
133             case 5: PORTDIGIPOTS = TREBLE_BOOST_CS_PIN;
134                     spiSend(0x00, val);
135             default: uputs("Invalid potmeter ID selected.");
136         }
137         PORTB = ~(1<<potno);
138
139     }
140
141     void toggleErrorLED(int onoff) {
142         if (onoff = 0) {
143             PORTLEDS = 0x00; /* Toggle error LED. CORRECT VALUE TO-BE-FILLED-IN */
144         }
145         else {
146             PORTLEDS = 0xFF; /* Toggle error LED. CORRECT VALUE TO-BE-FILLED-IN */
147         }
148
149     }
150
151     ISR (USART_RXC_vect) {
152         char c;
153         if (UCSRA & (1<<FE | 1<<DOR | 1<<PE)) {
154             c = UDR;
155             uputc('?');
156         }
157         else {
```

```
158             c = UDR;
159             /*PORTB = ~c;*/
160             if (rxcnt < 64) {
161                 sb[rxp & 63] = c;
162                 rxp = (rxp + 1) & 63;
163                 rxcnt++;
164             }
165             else {
166                 uputc('!');
167             }
168         }
169     }
170
171     /* Temporary function for PC debugging */
172     void uputc(char c) {
173         while (~UCSRA & 1<<UDRE);
174         UDR = c;
175     }
176
177     /* Temporary function for PC debugging */
178     void uputs(char str[]) {
179         int i;
180         for (i = 0; str[i] != '\0'; i++) {
181             uputc(str[i] | 0x80);       /* | 0x80 is debug code*/
182         }
183     }
184
185     char ugetc() {
186         char c;
187         int offset;
188         offset = (rxp - rxcnt) & 63;
189         c = sb[offset];
190         rxcnt--;
191         return c;
192     }
193
194
195
```