2015/2016

# Digital Implementation of a Motional Feedback Audio Sytem

Bachelor Graduation Project - EE3L11

Thesis

Delft University of Technology


**Project Team G**

| | |
|---|---|
| Ramon Overwater | 4305000 |
| Yann Rosema | 4218248 |

**As commissioned by:**
Dr. Ir. G.J.M.Janssen


Wednesday 15th June, 2016


TU Delft Delft University of Technology

Challenge the future

# Preface

This report is written as part of the Bachelor Graduation Project (BAP) for the Electrical Engineering bachelor at the TU Delft. The project group consists of six electrical engineering bachelor students, who are divided into subgroups of two and are each devoted to a subsystem of the total project.

The assignment was provided by Dr. Ir. G.J.M. Janssen out of interest in the digital implementation of the motional feedback system. Janssen provided further guidance during the project.

Other than Janssen we would like to thank the employees of the Tellegen Hall at the TU Delft for their help and the budget they provided. We also want to thank Ioan Lager for the coordination of the project. Lastly we want to thank the members of the jury for reviewing our work.

June 2016
*Oscar de Groot, Jaco Salentijn, Ramon Overwater, Yann Rosema, Wouther Bons, Tijs Hol*
TU Delft

# Abstract

Nowadays, the digital possibilities to implement control systems are easier to understand and results can be achieved quicker than in the 1970's, when Philips was experimenting with motional feedback speakers. This thesis covers the feedback system of a digital implementation of a motional feedback system for speakers. After the design choices are made an STM32F4-Discovery board with a Cortex M4 ARM MCU and a CS43L22 DAC with class D audio amplifier combined with a TI PCM1808 ADC are used to implement this digital support. Because the ADC could not be made operational in time, only results of the MCU and the DAC are be obtained and discussed. These results meet the requirements and show that if the ADC is made operational according to its specification, the design is valid.

# Contents

# List of variables

| | |
|---|---|
| $C$ | Capacitance |
| $f_mcu$ | Clock speed of the Microcontroller |
| $f_s$ | Sampling Frequency |
| $I_{add}$ | Amount of instructions it takes per add operation per order of numerator or denominator in the IIR filter |
| $I_{array\ shift}$ | Amount of instructions it takes per shift operation per order of numerator or denominator in the IIR filter |
| $I_{available}$ | Available instructions |
| $I_{multiply}$ | Amount of instructions it takes per multiply operation per order of numerator or denominator in the IIR filter |
| $I_{needed}$ | Instruction needed by the IIR filter |
| $M$ | Number of bits in word |
| $n$ | Order of the IIR filter |
| $Q$ | Quantisation Resolution |
| $R$ | Resistance |
| $SNR_Q$ | Quantisation Signal-to-Noise Ratio |
| $T_1$ | Time to integrate over integrator 1 |
| $T_2$ | Time to integrate over integrator 2 |
| $t_{adc}$ | Time it takes the ADC to convert an analog input to a digital output |
| $t_{dac}$ | Time it takes the DAC to convert an digital input to an analog output |
| $t_{io}$ | Time it takes for a signal to go through the ADC, MCU and DAC |
| $t_{io,max}$ | Maximum time it takes for a signal to go through the ADC, MCU and DAC |
| $t_{processing}$ | Time it takes the MCU to process the signal |
| $V_{Hi}$ | High Voltage |
| $V_{in}$ | Input Voltage |
| $V_{Low}$ | Low Voltage |
| $V_{ref}$ | Reference Voltage |

# List of abbreviations

| | |
|---|---|
| ADC | Analog-to-digital converter |
| AHB | AMBA High-performance Bus |
| AMBA | Advanced Microcontroller Bus Architecture |
| BCK | Bit clock |
| CPU | Central Processing Unit |
| DAC | Digital-to-analog converter |
| dB | Decibel |
| DC | Direct Current |
| DMA | Direct Memory Access |
| FMT | Format |
| FPGA | Field Programmable Gate Array |
| FSM | Finite State Machine |
| GPIO | General Purpose Input/Output |
| GPU | Graphic Processor Unit |
| GUI | Graphic User Interface |
| GND | Ground |
| HDL | Hardware Description Language |
| HSE | High Speed External Oscillator |
| HSI | High Speed Internal Oscillator |
| $I^2C$ | Inter Integrated Circuit |
| IIC | Inter Integrated Circuit |
| $I^2S$ | Inter IC Sound |
| IIR | Infinite Impulse Response |
| IIS | Inter IC Sound |
| L1 cache | First level cache memory |
| L2 cache | Second level cache memory |
| LRCK | Left-right clock |
| LSB | Least Significant Bit |
| MCK | Master clock |
| MCU | Micro Controller Unit |
| MD | Mode |
| MPU | Micro Processor Unit |
| PC | Personal Computer |
| PCB | Printed Circuit Board |
| PCM | Pulse Code Modulation |
| Pk-Pk | Peak to Peak |
| PLL | Phase Locked Loop |
| PLLM | Phase Locked Loop parameter M |
| PLLN | Phase Locked Loop parameter N |
| PLLP | Phase Locked Loop parameter P |
| PWM | Pulse Width Modulation |
| RAM | Random Access Memory |
| RMS | Root Mean Square |
| ROM | Read-Only Memory |
| SA ADC | Successive-Approximation ADC |
| SAR | Successive Approximation Register |
| SCK | Serial clock $I^2S$ |

| | |
|---|---|
| SCL | Serial clock $I^2C$ |
| SD | Serial data $I^2S$ |
| SDA | Serial data $I^2C$ |
| SNR | Signal-to-noise Ratio |
| $SNR_Q$ | Signal-to-noise Ratio due to Quantization error |
| SPI | Serial Peripheral Interface |
| THD | Total Harmonic Distortion |
| THD+N | Total Harmonic Distortion and Noise |
| UART | Universal Asynchronous Receiver/Transmitter |
| USB | Universal Serial Bus |
| $V_{cc}$ | Analog Voltage Supply |
| $V_{dd}$ | Digital Voltage Supply |
| WS | Word select |

# 1 Introduction

## Background[1]

### History of motional feedback

In the year 1968 Philips released a paper on motional feedback (MFB),[1] a technique where electro-acoustic feedback is used to reduce the linear and non-linear distortion of a speaker system. Philips and various other manufacturers sold audio systems with the motional feedback system for about a decade. Unfortunately the costs of the analogue equipment necessary for the implementation did not make the system worthwhile in the long run and it disappeared from the market.

Recent development in the costs and capabilities of digital devices have provided an opportunity for bringing back the motional feedback system by designing a digital implementation. The advantages of the modern techniques can make motional feedback an affordable extension to modern day audio systems.

### Principle of digital motional feedback

When playing sound over a loudspeaker, an electrical waveform resembling the sound is converted into air pressure waves which are sensed by human ears and perceived as sound by the brain. Because a loudspeaker is a non-ideal, physical system, this conversion from voltage to air pressure distorts the sound by for example damping out low frequencies (linear distortion) and adding harmonics that weren't present in the original waveform (non-linear distortion). Ideally, a loudspeaker would output the exact sound it receives at it's input, so finding a way to decrease this distortion is of interest.

One way to do so is by using a motional feedback system, where the output of the speaker is measured by a sensor and used to calculate the error in the input signal. This error signal is then fed back to the input (fig. 1a) to correct it. To calculate this error signal from the measured output, an LTI filter called the *controller* is designed using Control System theory so that it corrects the input of the loudspeaker, thereby altering the output sound to more closely resemble the input waveform.

Digital MFB implements this controller on a digital platform. This requires sampling of the loudspeaker's measured output signal and reconstruction of the controller's output signal to an analogue voltage for comparison with the input signal (fig. 1b).

## Division into subsystems[2]

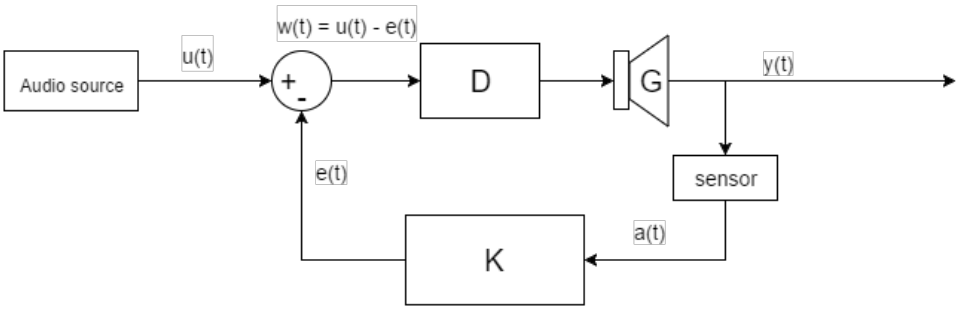
The design of the digital motional feedback device is split into three parts for the purpose of this project. The interfaces between these parts are defined as part of the Programme of Requirements (see chapter 2).

### Digital signal processing

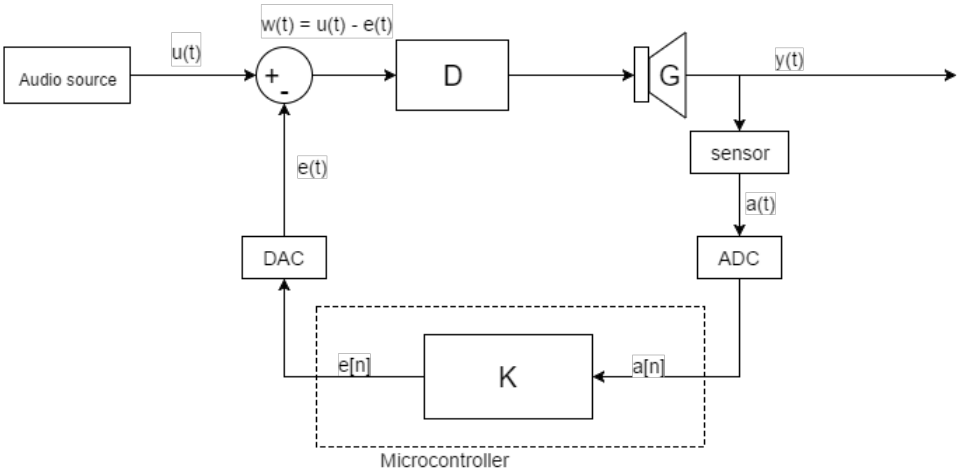
The first part is devoted to the digital signal processing, which is highlighted green in figure 2. The main tasks of this design are:

- Digital-Analogue conversion
- Analogue-Digital conversion
- Providing a configurable platform for the controller

[1] Because it discusses the project in general, this section occurs in the theses of all three subgroups (see section 1).
[2] Because it discusses the project in general, this section occurs in the theses of all three subgroups.




**(a)** Controller implemented as analogue system.




**(b)** Controller implemented digitally.

**Figure 1:** Layout of a motional feedback system around a loudspeaker system with speaker $G$ and amplifier $D$. Shown are the controller $K$, audio input signal $u(t)$, speaker output pressure $y(t)$, conus acceleration $a(t)$ and $a[n]$ and the error signal calculated by the controller $e(t)$ and $e[n]$.

The performance of this part is highly dependant on the delay in all respective stages.

This thesis will focus on the digital signal processing subsystem. For details on the design of the other two subsystems, please refer to [2] and [3].

**Analogue components**
The second part is the design of the analogue system components which are highlighted red in figure 2. The aim of this part is designing the motional feedback system for high fidelity applications. Fittingly the amplifier constructed uses active feedback to reduce output distortion. All tasks belonging to this group are:

- Subtracting the controller's error signal from the audio
- Cross-over filtering of the audio
- Amplifying the audio signal with minimal distortion
- Characterising the speaker
- Measuring the displacement of the loudspeaker cone

The design of this subsystem is documented in a different thesis. See [2] for details.

**Control and system identification**
The final group designs the controller highlighted in blue in figure 2 and implements system identification. The control group is tasked with implementing the motional feedback that makes the system closed loop stable and reduces distortion. All tasks of this group are:

- Designing the motional feedback controller
- Implementing system identification that identifies the second order characteristic of the speaker the motional feedback is applied to

Because the system contains system identification it can be applied to all enclosed speakers. This means no redesign is required for installing the motional feedback module on a different second order speaker system.

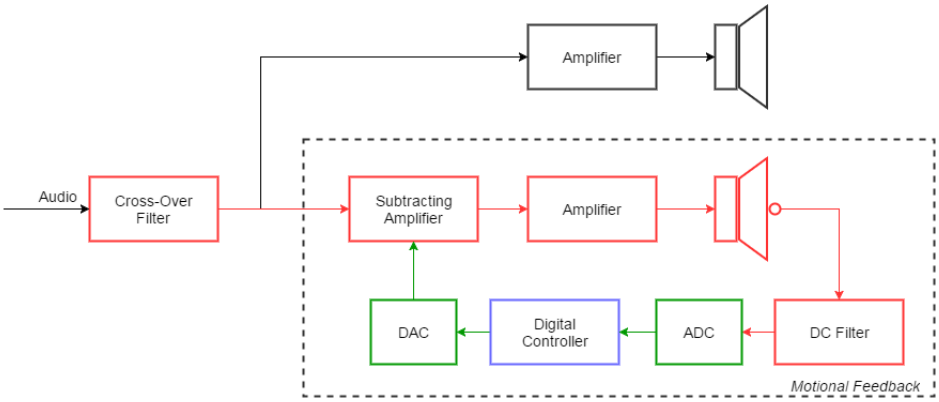



**Figure 2:** The motional feedback system designed by the three groups. The green parts are designed by the digital group. The parts highlighted in red are related to the amplifier/filter group. The blue part is the controller implemented by the control group. The MFB system is shown here in a two-way loudspeaker setup, where the tweeter (on top) does not use motional feedback.

## Focus of this thesis

This thesis covers the first part of listed subsystems, the digital signal processing. This thesis answers the need for data conversion and communication between the analog output of the speaker and the digital implementation of a control system.

Nowadays sound processing units are available on the market from professional tools for live gigs to devices for everyone at home. These units perform the task of digitising analog signals and processing it before making these signals analog again. For example, tools like the dbx DriveRack PA2, a loudspeaker management system for professional applications, which is used to enhance the quality of the output sound over a multi-speaker system. Although this tool is not present in a motional feedback loop, many of these tools, including this particular one, have the ability, using a precision microphone, to measure the frequency response of a speaker system to determine once the filter to be applied.
A lot of mid- to high end hi-fi and home cinema sets have a DSP inside to process the sound in some way. Usually this to compensate for the location of the listener or to simulate a certain kind of environment, but no device has an integrated digital motional feedback system in it.

One of the best attempts at bringing a digital motional feedback system on the market was Philips' DSS930 speakers [4]. These were speaker with a powerful integrated DSP that eliminated phase shift in the lower frequencies and compensated for impedance peaks.
Here at Delft University of Technology there has already done been research about MFB speakers as R. Valk did this master thesis on this subject [5]. He used a PC with a controller board (dSpace DS1103 PPC) to implement his project. This device however is mainly for research purposes, therefore other solutions need to be found to implement the link between the analog speakers and digital control system.

To best evaluate and answer the given problem a program of requirements is first set up in agreement with the project coordinator in section 2. Then the design of the system is presented in section 3, where the choices for the different tasks the system has to perform are made. The implementation of these choices and their results are then discussed in section 4. Afterwards, the results will be discussed in section 6 and finally the thesis will be concluded in section 5.

# 2 Program of requirements

This section covers the requirements of the system based on the state of the art analysis and limitations of the project itself. First the requirements of the total system are listed. These outline the overall purpose, the time and money restrictions, the input and output, and the criteria of the system transfer. Next the derived criteria for the subsystem designed in this thesis are listed.

1. **Total System**
   - 1.1. **Overall**
     - 1.1.1. *The feedback must be implemented digitally.*
     - 1.1.2. *The system must be able to log its input audio stream and controller parameters.*
     - 1.1.3. *The system must be adjustable to the speaker.*
     - 1.1.4. *The system must work on second order speaker systems.*
   - 1.2. **Input Output**
     - 1.2.1. *The maximum peak-to-peak input voltage of the audio signal is* $4\,V$.
     - 1.2.2. *The maximum output power is* $50\,W$.
     - 1.2.3. *The DC output may not exceed* $50\,mV$.
   - 1.3. **System Transfer**
     - 1.3.1. *The gain of the system must be* $28\,dB \pm 1\,dB$.
     - 1.3.2. *The THD must be inside the mask shown in appendix A.*
     - 1.3.3. *The bandwidth of the total system must be between 10 Hz and 20 kHz.*
     - 1.3.4. *The SNR is at least* $80\,dB$ *at the output inside the specified bandwidth of the motional feedback.*
     - 1.3.5. *The phase margin of the feedback transfer must be at least 45°to ensure stability.*
   - 1.4. **Development**
     - 1.4.1. *Development time is 9 weeks.*
     - 1.4.2. *Development costs must be below €300.*
2. **Digital System**
   - 2.1. **Overall**
     - 2.1.1. *The digital signal must be able to be processed by custom C code.*
     - 2.1.2. *The system must log its computed output as well as the model parameter to an external PC.*
     - 2.1.3. *The PC must be able to process the logged data to make a GUI.*
   - 2.2. **System Transfer**
     - 2.2.1. *Bandwidth between 10 Hz and 1 kHz.*
     - 2.2.2. *The phase shift between input and output of the feedback transfer must be smaller than* $135°$ *at maximum bandwidth to ensure a phase margin of 45°. Since the controller is allowed to have a phase shift of 90°the maximum allowed phase shift of the data conversion is 45°.*
     - 2.2.3. *The THD from the ADC, DAC and MCU combined may contribute to maximal 5% of the allowed THD by specification 1.3.2.*
     - 2.2.4. *The SNR must be larger than 100 dB.*
     - 2.2.5. *The* $SNR_Q$ *must be larger than the SNR.*
     - 2.2.6. *The THD and SNR are measured by outputting and inputting to a PC running Matlab at a sampling frequency of 32 kHz.*

### 2.3. **Input Output**

2.3.1. *Maximum Input Voltage Peak to Peak of 4 V.*

2.3.2. *Maximum Output Voltage Peak to Peak of 4 V.*

2.3.3. *Able to detect signals between 10 Hz and 1 kHz.*

2.3.4. *Able to output signals between 10 Hz and 1 kHz.*

2.3.5. *The sampling frequency must be at least four times the bandwidth (Two times Nyquist frequency).*

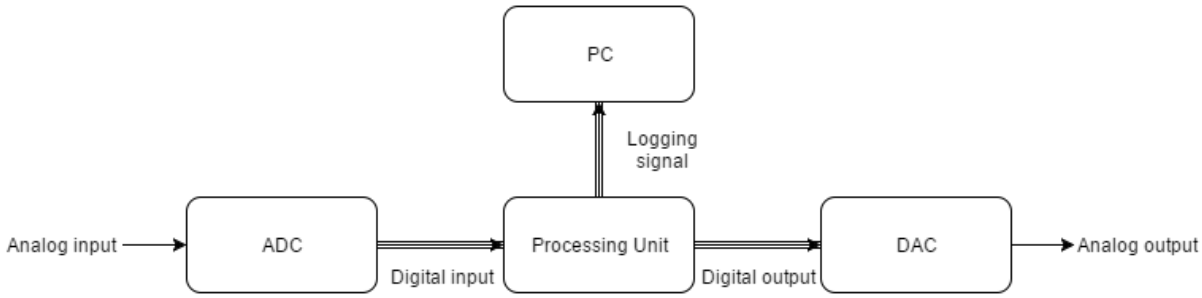



**Figure 3:** Overview of the system to be designed and implemented in this thesis.

# 3 Design

The full design of the system is shown in figure 2, this thesis covers the ADC, DAC and the support for a control system. These parts are specified in figure 3. Besides of the three parts, the communication method between the parts needs to be specified , this will be done in section 3.1 where the most commonly used communication protocols are discussed and compared to find which suits the application best. Then section 3.2 covers the options for ADCs and DACs. Section 3.3 looks at the differences between microprocessors, microcontrollers and FPGA's. Finally a preciser look is given to the sampling frequency in section 3.5.

## 3.1 Communication

Specifications require for communication between an ADC, a DAC, a processing unit and a PC. These communications are shown in figure 3. In this section a choice is made between parallel and serial communication. Next some serial communication protocols are described and compared.

### 3.1.1 Parallel vs Serial Communication

Parallel communication uses a wire for every bit of data as displayed in figure 4. This has the advantage of being able to send a single word per clock cycle, which has the possibility of being much faster than serial communication. The big disadvantage though is that you need a wire for every bit and that the GPIO of the processing unit also needs the same amount of inputs. Serial communication uses one wire for communication and sends one word as a stream of bits at a certain frequency. This frequency is either known by master and slave, or transmitted by the master. Since most serial communication protocols have speeds that are multiple times faster than needed for audio transmission, speed is not the biggest issue and the disadvantage of needing a lot more connections makes parallel communication a less appealing option.

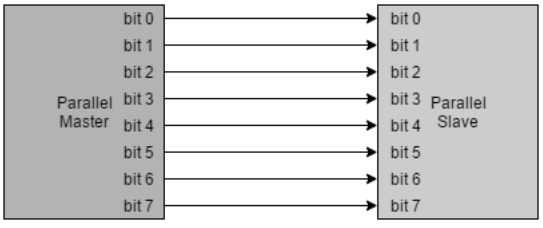



**Figure 4:** Master slave configuration using parallel communication

### 3.1.2 I$^2$C

I$^2$C or IIC [6] is a serial communication protocol designed by Phillips Semiconductors in the early 1980's. It was designed for speeds up to 100 kbit/sec, but nowadays reaches speeds of up to 3.4 mbit/sec. I$^2$C only needs two data lines, SDA and SCL, see figure 5. SDA is the data line and SCL the clock. Both lines are open-drain with a pull-up resistor at each terminal. This makes the lines high, until either the master or slave pulls the signal down. This allows for a couple of features, like multiple masters and slowing down the communication by the slave by stretching the SCL signal.


**Figure 5:** Master slave configuration using the I$^2$C protocol

I$^2$C usually has 7 bit addressing, which means that every device can have an address between 0 and 127. Communication is started by sending a start condition followed by the slave address and a bit to indicate whether the request is a write or read request. When the slave sees its address it sends an acknowledge bit back. This is shown in figure 6. Next the data is send until a stop condition is send.

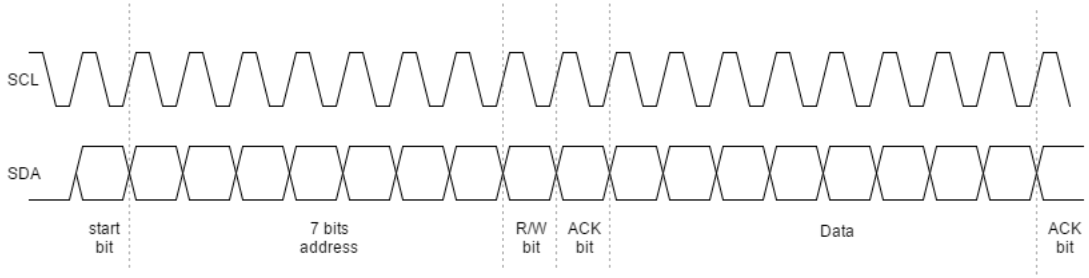



**Figure 6:** Sending one byte using the I$^2$C protocol.

### 3.1.3 SPI

SPI [8] is a synchronous serial communication protocol developed by Motorola. It uses a master-slave configuration with one master and multiple slaves. The master has one input and one output and a slave select output for every slave. This makes it possible for every slave to be on the same bus connected to the master, see figure 7. The master also creates a serial clock, connected to every slave. The communication is full-duplex, so every clock cycle the master sends a bit to the selected slave, and the selected slave sends a bit to the master. This is commonly implemented using a circular shift register as shown in figure 8

### 3.1.4 I$^2$S

I$^2$S or IIS [7] is despite its name unrelated to I$^2$C. It is however also developed by Phillips Semiconductors. It was designed as a serial communication stream for PCM audio signals. The protocol itself uses three data lines:

- WS or LRCK, this line has a frequency equal to the sampling frequency and selects whether the left or right audio channel is selected.

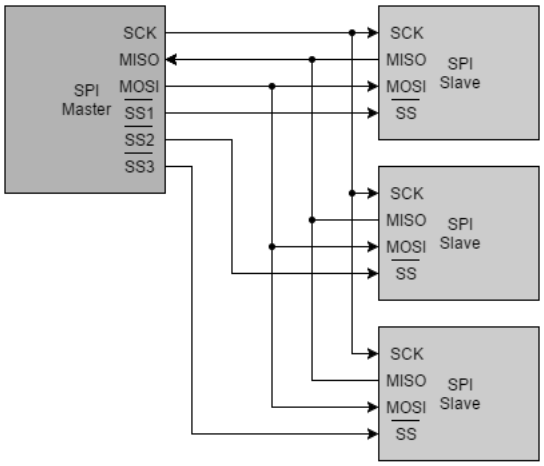



**Figure 7:** Master slave configuration using the SPI protocol with three slaves

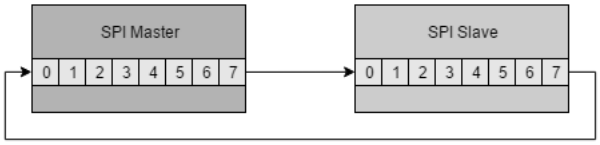



**Figure 8:** SPI master slave configuration using a circular shift register

- SCK or BCK, this line has a frequency of 32 or 64 times the sampling frequency depending on the word size and is used to indicate the bitrate.
- SD, this line has the data and updates at the same rate as the bit clock.

An example of an $I^2S$ signal is shown in figure 9.

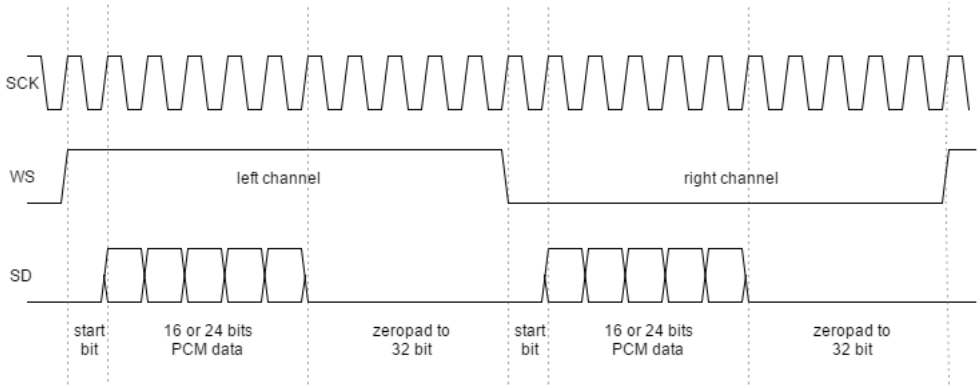



**Figure 9:** Sending a 16 or 24 bit stereo PCM audio signal over $I^2S$ using the standard Phillips $I^2S$ interface format.

Next to the three data lines, most slaves require a external system clock from the master, usually named MCK. The master is defined as the component that generates the WS and SCK signals as shown in figure 10. It is possible

that the master generates the WS and SCK signals and that the slave sends its data on the SD line.

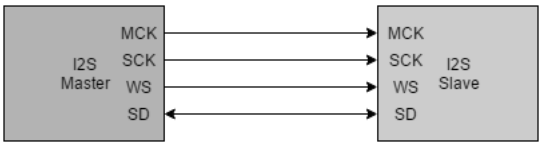



**Figure 10:** Master slave configuration using the $I^2S$ protocol

### 3.1.5 UART

UART [9] is one of the most basic forms of serial communication and is used in RS-232, RS-422, RS-485 and USB. When used on RS-232 or USB it has two separate lines for transmitting and receiving usually called tx and rx, see figure 11. Both lines are standard high and are pulled down by the transmitter for one bit as starting bit. Then one byte follows, followed by one or two ones as stop bits. An example of a UART signal is shown in figure 12.

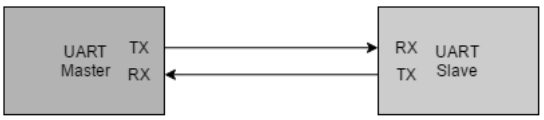



**Figure 11:** Master slave configuration using the UART protocol

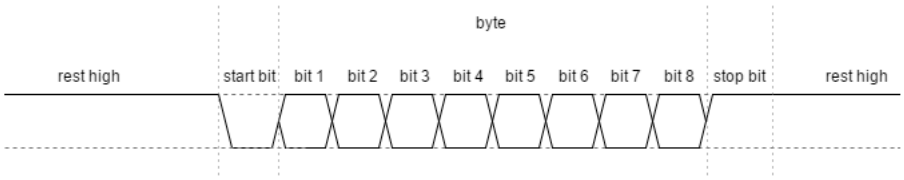



**Figure 12:** Sending one byte using the UART protocol

UART uses standardised baud rates. This means that both sides of the communication must be configured to the right baud rate. This is different from $I^2C$, SPI and $I^2S$, where the correct bit clock is send from the master to all the slaves.

### 3.1.6 Choices

There are three communication paths needed on the processor:

- Between the analog-to-digital converter and the processor
- Between the digital-to-analog converter and the processor
- Between the processor and the computer for data logging and visualisation

Since $I^2S$ is developed for audio streaming between processors and data converters, it is the standard communication protocol for most audio ADCs and DACs. Some ADCs and DACs also have an $I^2C$ or SPI port to configure

the ADC or DAC to the correct $I^2S$ settings. Others just use input pins connected to high or low to specify which $I^2S$ settings to use.
For communicating between the processor and a PC, UART is the most preferred choice, since the USB ports of a PC use the UART communication protocol.

## 3.2 Data Conversion

In order to use the data from the speaker sensor, which is an analog voltage, it needs to be converted to a digital bit stream. This job is done by an ADC. After the data has been manipulated it needs to be fed back into the system meaning the data needs to be analog again. Thus the bit stream should be converted back to an analog voltage. This is done by a DAC.
Both systems can be implemented in various manners which will be discussed in this section.

### 3.2.1 ADC

An analog to digital converter converts an analog voltage to a digital bit stream which is then sent using a communication protocol from section 3.1. Different aspects are of importance when choosing a type of ADC: resolution, conversion speed and step recovery.

The resolution of an ADC is dependent on the input voltage range and the number of digital output bits. Equation (1) gives the expression for the quantisation resolution. $V_{Hi} - V_{Low}$ is the input voltage range and $M$ is the number of output bits.

$$Q = \frac{V_{Hi} - V_{Low}}{2^M - 1} \tag{1}$$

A derivative characteristic of the resolution is the quantisation noise. This is the noise that occurs due to the fact that a digital number can not take all the values an analog signal can thus it rounds the analog value to the nearest digital value with a uniformly distributed error between $\pm 1/2$ LSB.
This error yield a signal-to-noise ratio due to quantisation given in equation (2).

$$SNR_Q = 20\log_{10}\left(2^Q\right) \tag{2}$$

The conversion speed is an important measure because it relates to the sampling frequency; an ADC with a high conversion speed will be able to sample the input with a high sample rate. If the sample rate is chosen too low for the input frequency aliasing may occur.
Aliasing is the effect which occurs when a signal is sampled in such a way that it can fit multiple signals or aliases as shown in figure 13.
To prevent aliasing the sampling frequency needs to be at least equal to two times the highest frequency that will be sampled, this is also known as the Nyquist frequency.

Step recovery is a measure of how fast a system can react to large sudden changes in the input data.

In the next paragraphs different types of ADCs will be discussed in order to determine which is most suited for the current application.

**Flash ADC** The flash is the simplest ADC in its operation. It is composed of a lot of comparators that compare the input voltage to a different reference voltage for each comparator. The output of all these comparators is then collected and encoded on a certain number of bits. An example of such an ADC is shown in figure 14.
This direct comparison allows this ADC to be very fast. But this speed has a cost which is the resolution. To get

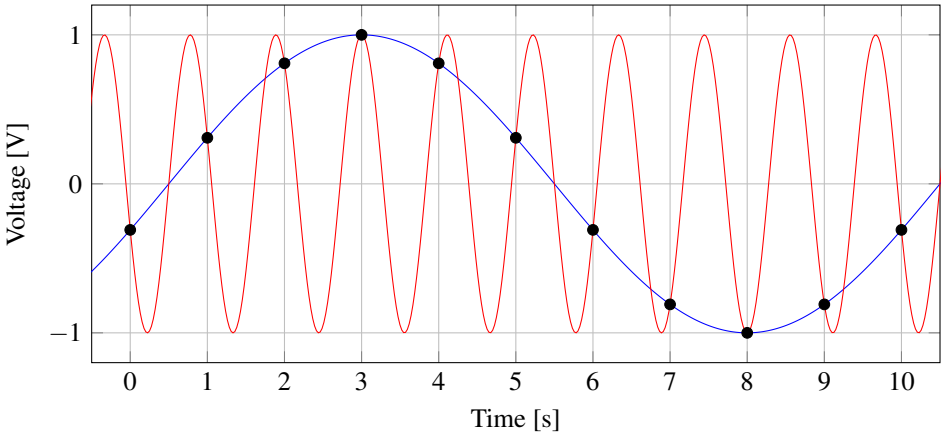



**Figure 13:** An example of aliasing using a red sine wave at 0.1 Hz a blue one at 0.9 Hz. The sample rate $F_s$ is 1 Hz. Both sine waves fit the same set of data shown as black dots.

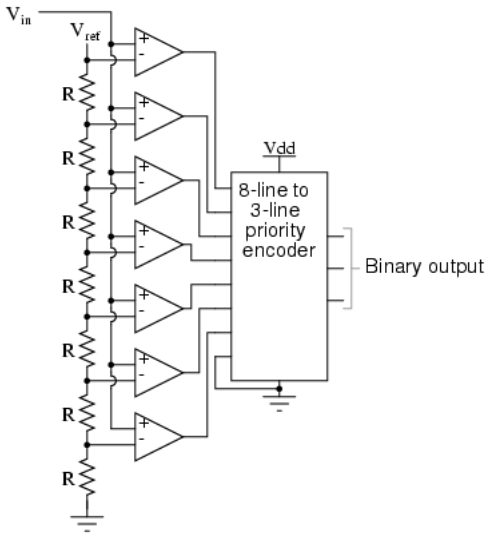



**Figure 14:** Diagram of a 3-bit flash ADC.

$N$ bits of resolution this ADC will need $2^N - 1$ comparators which will make the ADC big and expensive. For this reason, flash ADCs are used for low-resolution high-speed purposes.

**Digital ramp ADC** The way the digital ramp ADC works is also relatively simple. It is based on a counter that counts up on regular intervals. Then using a DAC, the output of the counter is compared with the input voltage. The output of the comparator is connected to the counter reset and the shift register enable input. When the counter value has just passed the analog value of the analog input the counter is reset and the output is set. After this, the counter starts again and the whole process restarts. A diagram of a digital ramp ADC is shown in figure 15.
Although this ADC is quite simple it has some big flaws. First of all, this ADC is slow because it needs to count all the way up to the analog input value. Second, because it counts with regular steps, low input values are reached earlier by the counter than high input values meaning that the samples are not spaced regularly. This flaw can been

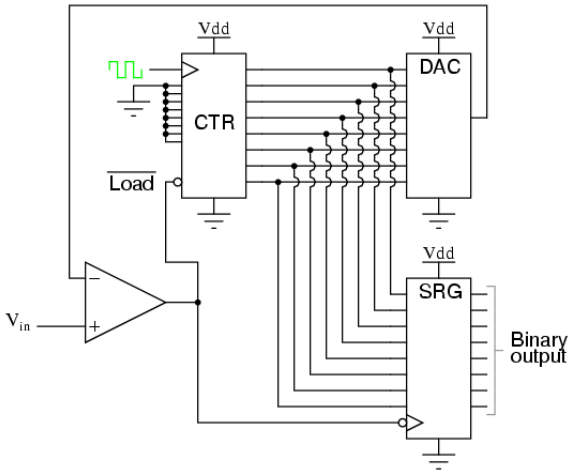



**Figure 15:** Diagram of an 8-bit digital ramp ADC.

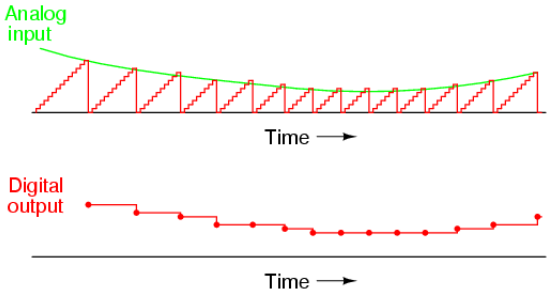



**Figure 16:** Plot of the input and output of a digital ramp ADC.

seen in figure 16.

**Successive approximation ADC** The successive approximation ADC uses a successive approximation register (SAR) to improve the performance of the digital ramp ADC. Where the digital ramp ADC counts from 0 to the full bit resolution, the SAR counts by filling the binary number the most significant bit (MSB) to the LSB. At each step the SAR makes, the output is compared with the analog input using a DAC. This output is fed back to the SAR which then puts the next bit to 1 or 0 according to the comparator output. This way the successive approximation ADC, approximates the input value a bit better each step. Figure 17 show the diagram of such an ADC.
The advantage of this way of converting an analog voltage to a digital value is that it is much faster than the digital ramp ADC. It also has the advantage of generating samples at regular interval unlike the digital ramp ADC. This is shown in figure 18.

**Tracking ADC** As the name suggests, the tracking ADC tries to track the analog input. It does this again by using a counter which output is compared with the input signal. The comparator switches the way the counter counts (up or down) to keep matching the input signal. This way the counter only needs to count all the way up to the signal only once and then switches the counter up or down. The diagram of this ADC is shown in figure 19.
Although it is a good regular way of sampling it has some drawbacks. First, as you might expect, the step recovery is very bad on this design because the counter has to count all the way to the new value of the input signal. Another

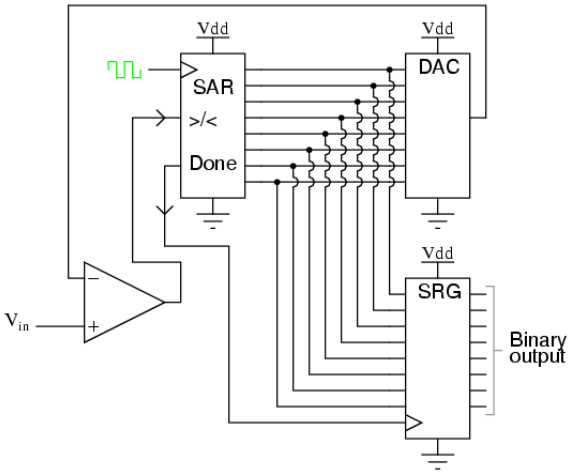



**Figure 17:** Diagram of an 8-bit successive-approximation ADC.

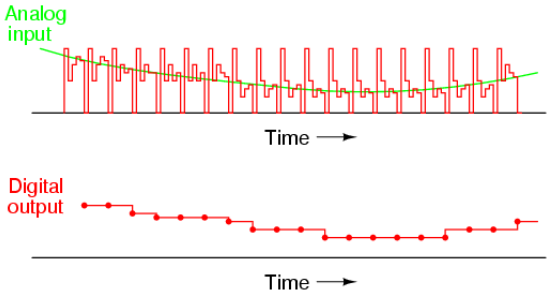



**Figure 18:** Plot of the input and output of a successive-approximation ADC.

drawback is the fact that the binary output is always changing. This is due to the fact that once the input signal is reached by the counter, it counts backwards but if the input signal is stable the new counter value will be too low thus the next step will increase the counter. Which again will be to high compared to the analog input and so on thus making the output vary each step. This can clearly be seen in figure 20.

**Slope (integrating) ADC** There are two variants of the slope, or integrating, ADC: the single- and dual-slope versions. We will first describe the single-slope ADC as the dual-slope is based on that one.
The single-slope ADC is similar to the digital ramp ADC in the way that it ramps up to the input voltage and then goes back to zero. But unlike the digital ramp ADC it does this with an integrator circuit. The time it takes to ramp up to the analog input is measured using a counter and send to the output using a shift register. The single-slope ADC has all the disadvantages of the digital ramp ADC plus one more which is calibration drift. This is the fact that over time the capacitor and counter will drift apart as the component ages and will therefore be imprecise. The solution to this problem is the dual-slope ADC shown in figure 21.
This version of the slope ADC is slightly different from the previous one. It uses an integrator too but with a counter activated switch in front of it. First the switch is set to integrate the input voltage, after a fixed time period set by a counter, the switch is flipped and the integrator integrates a fixed reference voltage of the opposite polarity as the input voltage. The counter now measure the time it takes for the integrator output to get back to 0 volt. An example of the integrator output is shown in figure 22.

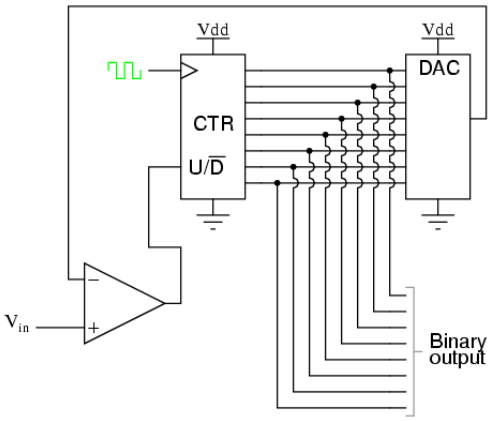



**Figure 19:** Diagram of an 8-bit tracking ADC.

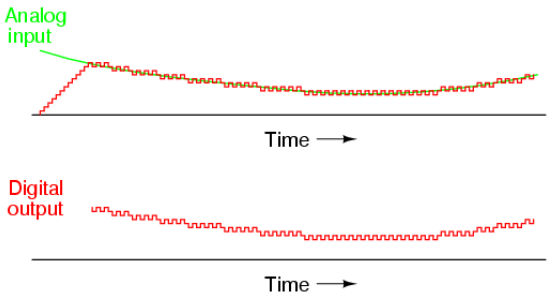



**Figure 20:** Plot of the input and output of a tracking ADC.

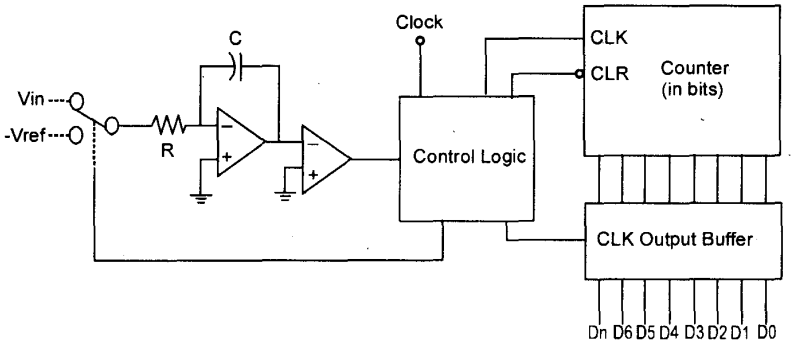



**Figure 21:** Diagram of a dual-slope ADC.

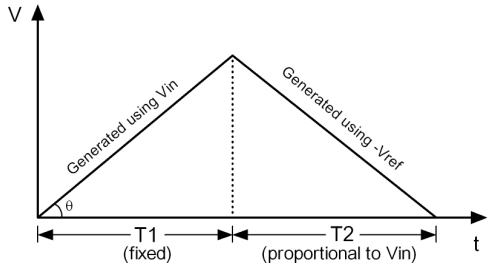



**Figure 22:** An example of the integrator output of a dual-slope ADC.

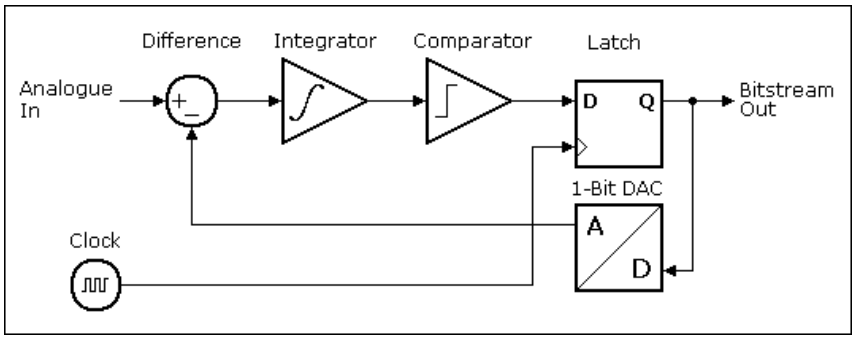



**Figure 23:** Diagram of a delta-sigma ADC.

Using equation (3) (with $V_{ref}$ the reference voltage, $V_{in}$, the input voltage, $R$ and $C$ respectively the resistor and capacitor value of the integrator and $T_1$ and $T_2$ respectively the fixed time to integrate $V_{in}$ and the measured time for the output of the integrator to get back to 0 volt) we can calculate the undefined input voltage.

$$\frac{V_{in}}{RC}T_1 = -\frac{V_{ref}}{RC}T_2 \tag{3}$$

Which simplifies to equation (4).

$$V_{in} = -V_{ref}\frac{T_2}{T_1} \tag{4}$$

With this design, the drift is literally eliminated from the equation as nor the capacitor neither the resistor value appears in the equation and the clock is used to produce a ratio, meaning that if the clock signal changes while the component ages, the ratio is not affected.
One big drawback of the ADC is the resolution which is limited by the integrator op-amp, most dual-slope ADCs have a resolution of less than 8 bits.

**Delta-Sigma ($\Delta\Sigma$) ADC** The delta-sigma ADC is probably the most complex ADC of the ones presented here. It is composed of a 4-stage feedback loop: first, the integrator stage, then a comparator stage followed by a latch and an DAC to finish (see figure 23). The system works as follows: the input voltage is subtracted from the DAC output which switches only between positive and negative maximum voltages. This signal then goes to the integrator, which is then fed to the comparator. Next the latch, driven by a high speed clock switches the input of the DAC.
An example of how the system works is shown in figure 24.

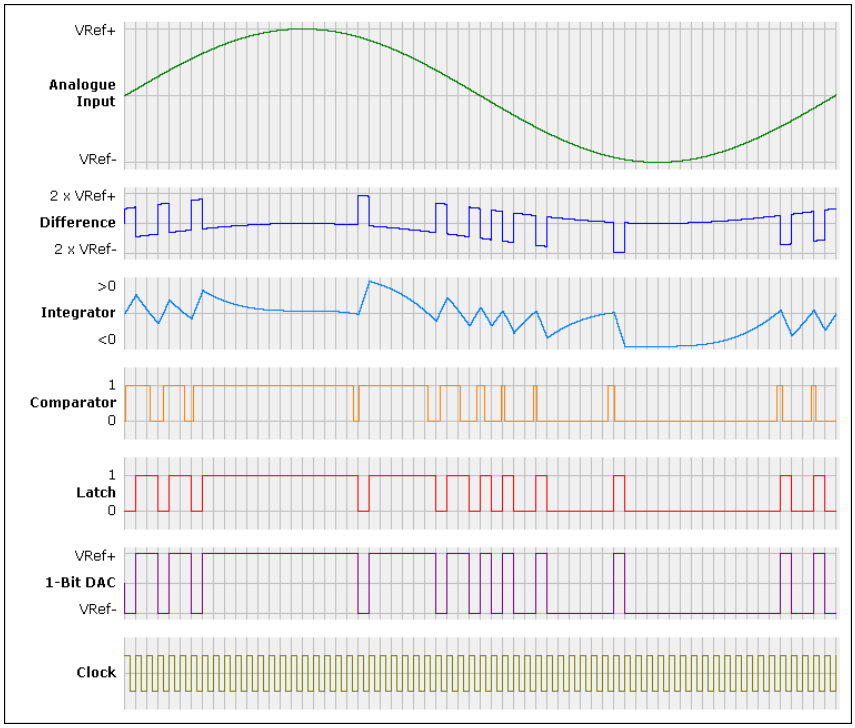



**Figure 24:** An example of the working of a delta-sigma ADC over time.

An advantage of this ADC is the fact that it is very easy to use oversampling with this data. A simple decimation filter can generate a higher resolution output than the 1-bit stream it normally generates. This also allows for great noise reduction when the clock frequency is much higher than the sample input.

**Choices** According to the program of requirements [2] there are specifications for the error from the resolution of the conversion and for the sample frequency. The former should exceed the SNR and the latter should be at least 4 times the bandwidth which sits between 10 Hz and 1 kHz. This means that the $SNR_Q$ should be greater than 100 dB and that the sample frequency should not be smaller than 4 kHz.
For the signal-to-digital-noise-ratio we can use equation (2) to find that $Q > 16,6$ or, because $Q$ is an integer $Q > 17$. This first condition already eliminates the flash and the slope ADC which are rarely made for application requiring more than 8 bits. The biggest flaw of the digital ramp ADC is its irregular sample rate which is not wanted here because the data signal will be audio, the variation between maximum and minimum input voltage are to big to have a regular output from the digital ramp ADC. The tracking ADC is not an option either as they are very uncommon and can not be found with more than 12 bit resolution. Successive-approximation ADCs and the delta-sigma ADC's can both be found with enough resolution and with a high enough sample frequency; to make a decision between these two ADCs another parameter comes into play: costs; a SA ADC is more expensive to make than a $\Delta\Sigma$ ADC. For this last reason the decision has been made to choose a $\Delta\Sigma$ ADC. To make sure to have enough room for error, especially for error in noise generation, the chosen ADC is a 24-bit delta-sigma ADC with a maximum sample frequency of 192 kHz.

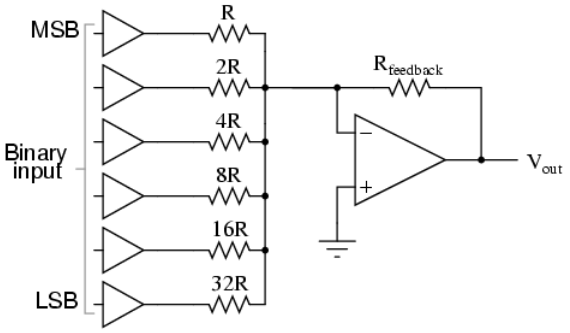



**Figure 25:** Diagram of a binary weighted DAC.

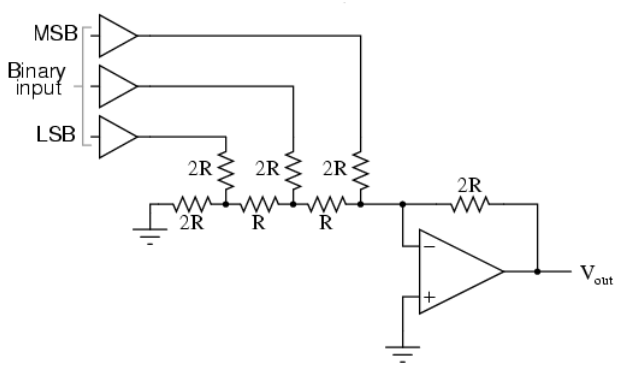



**Figure 26:** Diagram of a R/2R DAC.

### 3.2.2 DAC

After the data has been digitally processed it needs to be converted back to an analog value, this the job of a DAC. The same three aspects covered for the ADC are of importance for the DAC as well; in the last paragraph a decision will be made using these criteria.

**Binary Weighted DAC** The binary weighted DAC is, together with the R/2R DAC, maybe the most easy to understand. The binary weighted DAC works by summing the current through a resistor with a different weight for each bit of the ADC. This current then goes through a feedback resistor to generate the right analog voltage. A diagram of this DAC is shown in figure 25.
This system has, like the flash ADC, one big flaw and that is the number of components as the inputs need to be buffered too in order to make the voltages stable add the input of the summer. Because of this, binary weighted DACs are usually limited to 8 bit resolution.

**R/2R DAC** The R/2R DAC is similar to the binary weighted DAC in that they use the same components, but this time the binary input is arranged on a resistor ladder so that there are only two resistor values needed which is easier to produce. This can be seen in figure 26.
Unfortunately this does not eliminate the flaw of the binary weighted DAC that it needs a lot of components and thus is not used for more than 8 bit applications. Some 24 bit r/2R DACs exist, like the PCM 1704, but these are very expensive and therefore out of reach for the application.

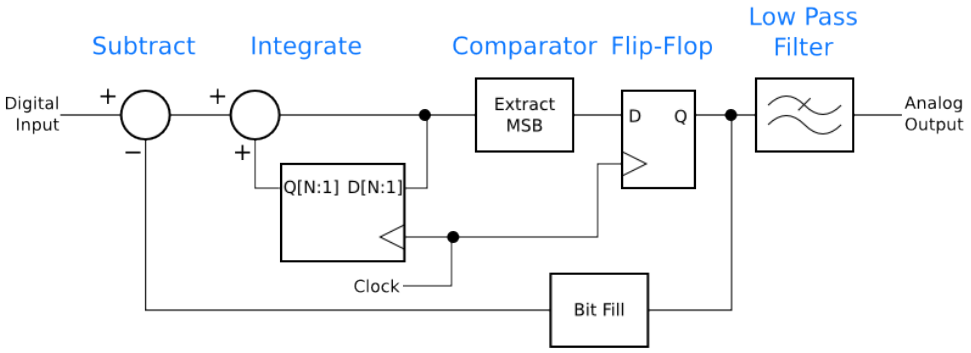



**Figure 27:** Diagram of a delta-sigma DAC.

**PWM** PWM stands for pulse width modulation and works as follows. A square signal is generate with the time the square signal is high that varies according to the binary input. This square wave is then filtered using a low pass filter too generate a continuous wave.

**Oversampling DAC (ΔΣ)** The delta-sigma DAC works almost as a delta-sigma ADC in reverse order. Where the ADC was down sampled at the last step, here we first interpolate the digital signal using an interpolation filter, which is than sent to a delta-sigma structure like in the ADC with an integrator, or adder in this digital case, adding the new input to the previous one stored in a register. This signal then goes to a digital comparator which switches a latch. The output of the latch is then extended with ones if the latch output is 1 or with zeros if the latch output is 0 and subtracted from the input of the DAC. The output of the DAC is generated at the end of a low pass filter which input is the latch output. The full diagram of this DAC is shown in figure 27.

**Choices** As explained earlier, the minimum resolution is 19 bits and the sample frequency should be a least 4 kHz. This means that both the binary weighted and R/2R DACs are not usable or this application. PWM DACs are not suited either also because of their low resolution, usually their resolution has a maximum of 16 bits. This leaves only the delta-sigma DAC as an option. This ADC is great for this application because it can fit all the criteria and, like its ADC counter part, is cheap to make.

## 3.3 Processing

Between the ADC and the DAC the data needs to be processed according to a control system. This control system is not part of this thesis, however the possibility for the control system to be implemented is part of it.
The program of requirements specifies two requirements for the processing system: it has to support C-code implementation 2.1.1. and the output the processing unit sends to the DAC as well as the model parameters have to be available for a PC to read out in order to create a GUI 2.1.2., 2.1.3.. This GUI will allow the user to monitor how the processing unit works or event manipulate the model parameters if they feel the need to do so.
In this section, the considered options for the controller implementation are explained and weighted against the requirements listed above.

### 3.3.1 FPGA

An FPGA (short for Field Programmable Gate Array) is a user programmable integrated circuit made of configurable logic gates. These devices use logic gates connected with a hardware description language (HDL) to configure any program an application-specific integrated circuit (like the ADCs and DACs listed above) can do. A

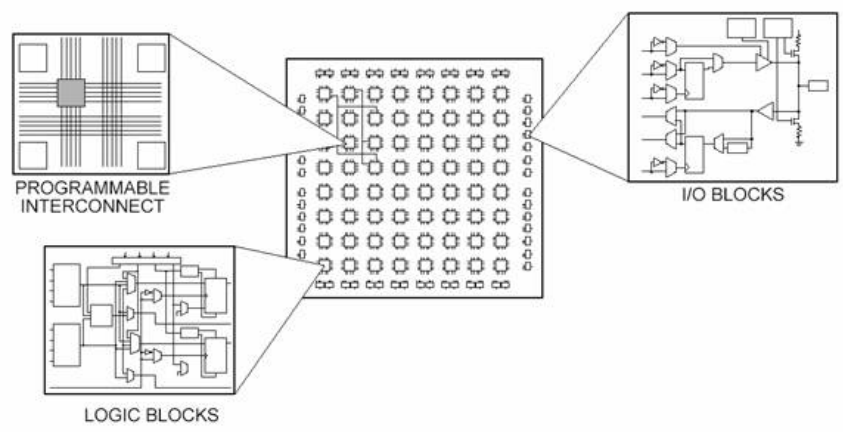



**Figure 28:** Diagram of the composition of an FGPA.

diagram of an FPGA is shown in figure 28. The main advantage of an FPGA is its concurrent way of handling the programmed circuits. Usually one would program finite state machines (FSM) that run simultaneously on the FPGA. This is the opposite of sequential operations which execute one after each other. Another advantage of the FPGA is its flexibility in output and input ports; every port can be any input or any output, which makes it easy when designing a PCB.
Unfortunately, as one of the requirements is the support of C programming, which is a sequential programming language, an FPGA would have to contort itself to work that way and thus the benefits of having an FPGA would get lost.

### 3.3.2 MPU

An MPU (short for Micro Processor Unit) is a programmable micro computer with one or more processors. Such system can be found in any computer, in this case the one used in a Raspberry Pi 2 was considered. An MPU is, as mentioned above, composed of one or more processors (4 in the case of the Raspberry Pi 2) but also L1 and L2 caches and a GPU. Like a computer, an MPU works with an operating system but it can also be programmed with C-code, which is called "bare metal programming". The biggest advantage of an MPU is perhaps the support for an OS (operating system) which generates a pre programmed environment to work in.
However, because the requirements specifically ask for the support of C-code, which would require bare metal programming, the MPU is not suited for this application. Bare metal programming is possible but requires a lot of research and overhead programming to be sorted out and thus would not be possible to be realised given the time period given. Moreover, the MPU has not got RAM nor ROM to store data, so these would have to be stored on a separate chip increasing design complexity. But maybe the biggest drawback of the MPU is the latency this system has. An early test with an OS driven audio processing software yielded a delay of 1.7 ms which is more than a full period delay at the top of the bandwidth. This would lead to instability of the control system.
For these reasons, the MPU was not chosen for the implementation of this application.

### 3.3.3 MCU

An MCU (short for microcontroller Unit) is a simplified version of the the MPU. The MCU contains a controller, to execute the programmed instruction, RAM, ROM and programmable input/output peripherals. The most commonly used programming language for these MCUs is C-code or C++, depending on the architecture of the chip. A suited compiler than compiles the high level code to machine code which is then uploaded to the chip. This is probably the best advantage of this device; its ability to easily implement high level code such as C-code. Another

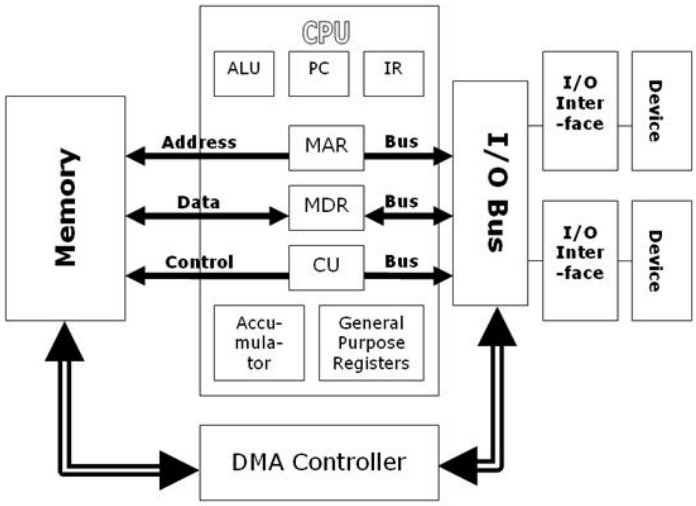



**Figure 29:** Diagram of an MPU with DMA.

advantage of this device is that most of them already have build in support for the communication protocols mentioned in section 3.1. This simplifies programming as the protocols do not need to be written but only the startup code and the to-be-transmitted data need to be provided.
For these reasons the chosen device to implement the control system is an MCU.

## 3.4 DMA

Direct Memory Access is a principle found in MPUs and MCUs that allows peripherals to have direct access to the data memory of the unit without using the CPU. The DMA controller, which controls the flow of data from and to the memory, makes a direct connection between the in- and output peripherals and the memory. Figure 29 shows the connections of DMA in an MPU.
By skipping the CPU in this read and/or write operation from and/or to the memory, the CPU has more computing power for manipulations on the data. Once a DMA stream is set, it runs on the background until the stream is set to stop.

## 3.5 Sampling Frequency

As required from specification 2.2.2. the phase shift must be smaller then 45°at maximum bandwidth. This gives limitations to the sampling frequency. The time it takes for an input sample to reach the output is the sum of three variables as shown in equation 5, where $t_{adc}$ is the time it takes for an analog signal to reach the processor, $t_{dac}$ the time it takes for an digital output from the processor to be outputted from the DAC and $t_{processing}$ the time it takes for the controller to compute the output.

$$t_{io} = t_{adc} + t_{processing} + t_{dac} \tag{5}$$

The maximum frequency in the feedback loop is 1 kHz. This means that the maximum time that $t_{io}$ can be is:

$$t_{io,max} = \frac{45°}{360°} \cdot \frac{1}{1\,\text{kHz}} = 125\mu\text{s} \tag{6}$$

The time it takes for the ADC and DAC for converting their input data when using $I^2S$, is one cycle of their WS clock. And since the WS clock is at the sampling frequency $f_s$, this gives $t_{adc} = t_{dac} = \frac{1}{f_s}$. If the time the processing takes is also limited to $t_{processing} = \frac{1}{f_s}$, the pipe-lining structure shown in figure 30 is obtained.

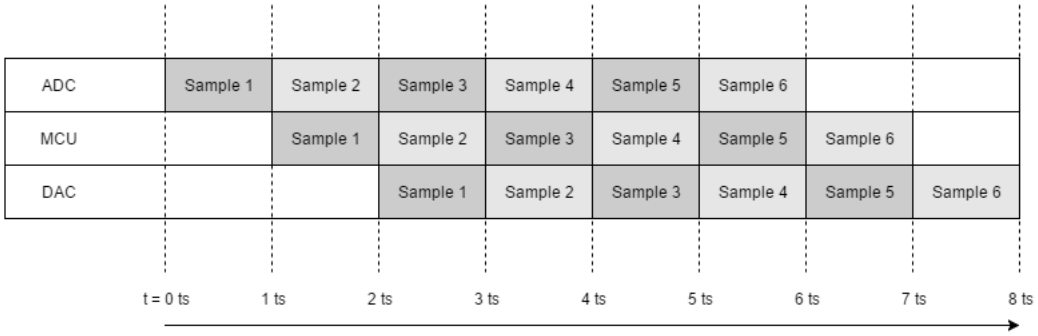



**Figure 30:** The pipe-lining structure of the input and output data through the ADC, MCU and DAC.

Combining that $t_{adc} = t_{dac} = t_{processing} = \frac{1}{f_s}$ with equation 5 and 6, the result in equation 7 is found.

$$3 \cdot \frac{1}{f_s} \leq t_{io,max} \text{ or } f_s \geq 24\text{kHz} \tag{7}$$

This is quite a lot higher than the minimum required sampling frequency of 4 kHz from the requirements. It is also important to keep the sampling frequency as low as possible, so the controller system has as much time possible to do its calculations.
To see if the controller has enough time, a worst case approximation is made. The following assumptions are made for the approximation:

- An IIR filter of order $n = 20$ in denominator and nominator.
- The MCU has a clock speed ($f_{mcu}$) of 168 MHz.
- The overhead, interrupts and communications take up 80% of the clock cycles, this leave 20% for the controller to use.
- The sampling frequency ($f_s$) is 32 kHz, this is higher than 24 kHz to take into account the possibility that 24 kHz is no possible sampling frequency for either the ADC, DAC or MPU.

With an IIR filter, every order of the denominator and nominator has to be multiplied by a factor. This takes up four instructions. Two to load the floats from the sample and the filter, one to multiply and one to save the result. Next every order of the denominator and nominator, except the first one, has to be added to the rest. Adding also takes up four instruction following the same logic as with multiplying. Finally the samples need to be shifted in the array. This takes four instructions, one to load and one to save and two to calculate the indices of the array. The amount of instructions needed ($I_{needed}$) is shown in equation 8. The denominator and nominator have the same order, a factor of two is used to add the two.

$$I_{needed} = 2 \cdot (I_{multiply} \cdot n + I_{add} \cdot (n-1) + I_{array\,shift} \cdot n) = 472 \tag{8}$$

The amount of available instructions ($I_{available}$) is shown in equation 9.

$$I_{available} = \frac{f_{mcu} \cdot 20\%}{f_s} = 1050 \tag{9}$$

Since the controller has enough time in the worst case, it will also work if the overhead takes up less instructions and the IIR filter is of lower order.

# 4 Implementation

## 4.1 Hardware

From section 3.3 the best option for a processing unit was found to be an MCU. It is also necessary for the MCU to have float support. Finally it would be preferable if there was a development board that features a compatible MCU with a DAC and/or ADC that meet the requirements. The STM32F407VG [10] is such a development board. It contains a Cortex M4 ARM microcontroller [11] and a Cirrus Logic audio DAC with class D amplifier [14]. This makes it so the DAC can deliver enough power to subtract it from the audio input without additional amplifiers. This leaves the ADC. The TI PCM1808 [15] is a cheap and sufficient solution, with the benefit that it needs no other communication than $I^2S$.

## 4.2 HAL Drivers

The HAL Drivers [12] are an API between the low level peripherals of the STM32 series and the programmer, so it is compatible with all boards without having to change all the specific MCU registers. STM has also created software that creates C code using a GUI called STM32CubeMX.

### 4.2.1 Clock Configuration

Without any configuration the clock of the MCU runs on an internal RC resonator with a clock speed of 16 MHz. This however gives a rather distorted clock signal which is also used to drive the $I^2S$ peripheral and is visible on the audio output. For this reason a quartz oscillator of 8 MHz is used as external clock. To create the maximum clock speed of 168 MHz a Phase Locked Loop or PLL is used. The formula for the PLL clock of the STM32 is shown in equation 10, where HSE is the external clock speed and PLLN, PLLM and PLLP parameters for the PLL.

$$\text{SYSCLK} = \text{HSE} \cdot \frac{\text{PLLN}}{\text{PLLM} \cdot \text{PLLP}} \tag{10}$$

Then there are a couple of peripherals that are scaled down using AHB prescalers. Which peripherals depend on clock 1 and 2 are different per board, see [13].

$$\text{ETHERNET CLK} = \frac{\text{SYSCLK}}{\text{AHB}}$$

$$\text{PERIPHERAL 1 CLK} = \frac{\text{SYSCLK}}{\text{AHB} \cdot \text{AHB1}}$$

$$\text{PERIHPERAL 2 CLK} = \frac{\text{SYSCLK}}{\text{AHB} \cdot \text{AHB2}}$$

The clock of the UART must be 48 MHz and is defined using equation 11.

$$\text{UART CLK} = \text{HSE} \cdot \frac{\text{PLLN}}{\text{PLLM} \cdot \text{PLLQ}} \tag{11}$$

Finally the $I^2S$ clock needs a clock of 51.2 MHz and is defined by equation 12.

$$\text{I2S CLK} = \text{HSE} \cdot \frac{\text{PLLI2SN}}{\text{PLLM} \cdot \text{PLLI2SR}} \tag{12}$$

Solving for all the required clock speeds the correct parameters are displayed in table 2

**Table 2:** Parameters for the clock configuration

| Parameter | Value |
|---|---|
| PLLN | 336 |
| PLLM | 8 |
| PLLP | 2 |
| PLLQ | 7 |
| AHB | 1 |
| AHB1 | 4 |
| AHB2 | 2 |
| PLLI2SN | 256 |
| PLLI2SR | 5 |

### 4.2.2 $I^2C$

The $I^2C$ interface is needed to configure the Digital to Analog Converter. As mentioned in section 3.1.2 the GPIO pins need to be set to pull up open drain. The SDA line is connected to GPIO B9 and SCL to GPIO B6. So these pins needs to be configured to be open drain, pull up and connected to the $I^2C$ peripheral which is at alternate function 4. The configuration for the $I^2C$ peripheral itself is shown in table 3.

**Table 3:** Parameters for the $I^2C$ configuration

| Parameter | Value |
|---|---|
| Instance | I2C1 |
| Clock Speed | 100 kHz |
| Addressing | 7 bit |
| Slave Address | 148 |
| Clock Stretch | No Clock Stretching |
| Dual Address Mode | Disabled |
| General Call Mode | Disabled |

### 4.2.3 $I^2S$

There are two $I^2S$ peripherals used, the DAC is connected to $I^2S3$ and the ADC to $I^2S2$. Table 4 shows the GPIO connections to both peripherals. On the STM boards, the $I^2S$ peripherals are positioned on the SPI bus with the corresponding instance number. The initialisation parameters of the $I^2S$ peripherals and the DMA stream they are connected to are shown in table 5 and 6. The sampling frequency is the lowest possible value above the minimum required sampling frequency as calculated in section 3.5. Also the DAC is configured as slave and the ADC as master. This is due to the required input voltages of the ADC explained in section 4.4. Every output pin is configured as pull-up and every input pin as no pull.

### 4.2.4 UART

The UART4 peripheral has pin TX and RX on respectively GPIO A0 and A1. Both pins are Push-Pull Pull-Up. The initialisation configuration is displayed in table 7. To minimise the time that the logging takes, the highest baud rate is chosen.

**Table 4:** $I^2S$ GPIO connections and configuration

| | $I^2S2$ (AF5) | | $I^2S3$ (AF6) | |
|---|---|---|---|---|
| **Pin Name** | **GPIO** | **Config** | **GPIO** | **Config** |
| MCK | C6 | Push-Pull Pull-Up | C7 | Push-Pull Pull-Up |
| SCK | B10 | Push-Pull No-Pull | C10 | Push-Pull Pull-Up |
| WS | B12 | Push-Pull No-Pull | A4 | Push-Pull Pull-Up |
| SD | C3 | Push-Pull No-Pull | C12 | Push-Pull Pull-Up |

**Table 5:** Parameters for the $I^2S$ configuration

| **Parameter** | **$I^2S2$** | **$I^2S3$** |
|---|---|---|
| Instance | SPI2 | SPI3 |
| Mode | Slave RX | Master TX |
| Standard | Philips | Philips |
| Data Format | 24 bits | 24 bits |
| MCLK Output | Enabled | Enabled |
| Audio Frequency | 32 kHz | 32 kHz |
| Clock Polarity | Low | Low |
| Clock Source | I2S PLL Clock | I2S PLL Clock |
| Full Duplex Mode | Disabled | Disabled |

## 4.3 Configuring DAC

The power up sequence of the CS43L22 [14] is listed as follows:

1. Reset until power supply is stable.
2. Keep the Power Control 1 Register (0x02) in Power Down (0x01).
3. Initialisation sequence.
   - 3.1. Write 0x99 to register 0x00.
   - 3.2. Write 0x80 to register 0x47.
   - 3.3. Write '1'b to bit 7 in register 0x32.
   - 3.4. Write '0'b to bit 7 in register 0x32.
   - 3.5. Write 0x00 to register 0x00.
4. Apply MCLK. SCLK may be set any time and LRCK only when powered on.
5. Set Power Control 1 Register (0x02) in Power Up (0x9E).

Custom settings must be applied after step 3. The first setting is to set the headphone channels always on, and the speaker channels always off. This is done by setting register 0x04 to 0xAF. Next the headphone gain is set. This is done by writing to register 0x0D. To set both headphones to an output voltage of 2 V Peak to Peak, this register should be set to 0x70. For clocking control to auto detect the clocks, register 0x05 should be set to 0x81. Register 0x06, configures the interface. To set the DAC as a slave, with non inverted SCLK polarity, Phillips standard 24 bit, this register should be set to 0x07. To disable zero crossings and soft ramps, register 0x0A has to be set to 0x00. The volume of channel A and B are set by register 0x1A and 0x1B respectively, where 0 dB is 0x00.

**Table 6:** $I^2S$ DMA configuration

| Parameter | $I^2S2$ | $I^2S3$ |
|---|---|---|
| Instance | Stream 3 | Stream 5 |
| Channel | 0 | 0 |
| Direction | Peripheral to Memory | Memory to Peripheral |
| Peripheral Increase | Disabled | Disabled |
| Memory Increase | Enabled | Enabled |
| Peripheral Data Alignment | Half Word | Half Word |
| Memory Data Alignment | Half Word | Half Word |
| Mode | Normal | Normal |
| FIFO Mode | Disabled | Disabled |

**Table 7:** The configuration of the UART peripheral

| Parameter | Value |
|---|---|
| Instance | UART4 |
| Baudrate | 921600 |
| Word size | 8 bit |
| Stopbits | 1 |
| Parity | None |
| Mode | TX and RX |
| Hardware Flow Control | Disabled |
| Oversampling | 16 times |

## 4.4 Configuring ADC

The PCM1808 [15] has four operating modes, controlled by the two MD pins:

- 00: Slave mode, with auto detection of $f_s$ at MCK frequencies of $256f_s$, $384f_s$ and $512f_s$.
- 01: Master mode, with MCK at $256f_s$.
- 10: Master mode, with MCK at $384f_s$.
- 11: Master mode, with MCK at $512f_s$.

The ADC needs the voltages of SCK and WS at GND or at $V_{dd}$ which is 3V. The MCU however outputs a voltage peak-peak of 5V to 6V. One way to get around this problem is to use a voltage follower with a voltage divider to get a 3V signal. Another way is to use the ADC as master. This way the MCU can be directly connected to the ADC. Since the ADC is used as a master and the MCU creates an MCK at $256f_s$, MD0 has to pulled to $V_{cc}$ and MD1 has to be pulled to ground. The pin to select the format, FMT, has to be set to the standard Philips $I^2S$ format. This means that it also has to be pulled to ground.

## 4.5 Logging Test

The logging is tested by putting a 24 bit resolution signal on the ROM of the MCU and then sending it over UART to a PC running MatLab. MatLab then fills an input buffer of the same size as the number of samples in the signal. The signals are then compared. The used signal is a 20 Hz sine wave with the first four harmonics. This signal is first set to a unsigned 24 bit integers. The results are discussed in the next section.

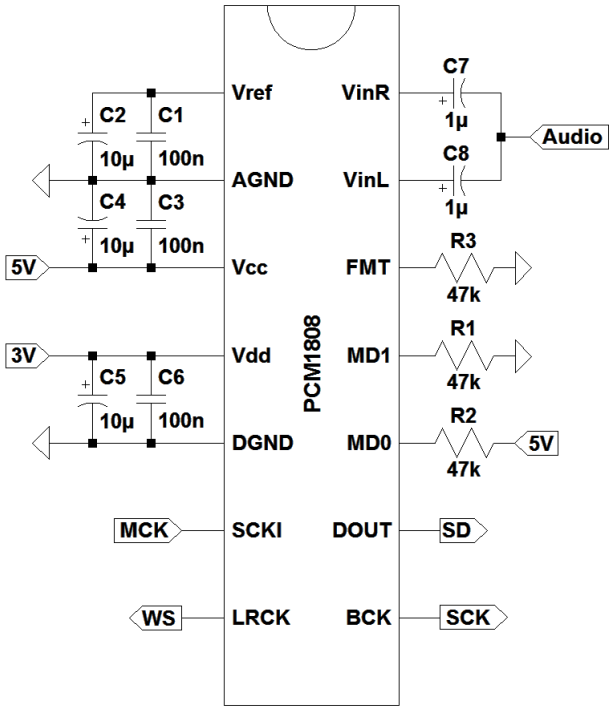



**Figure 31:** Schematic of the connections of the PCM1808.

### 4.5.1 Results

In figure 32 the measured signal is displayed. The top left plot contains the original signal of 5 sines in two's complement notation while the top right plot contains the signal as unsigned integers. The two bottom plots contain the received signal from both channels. Despite the fact that they are translated in time it is quite clear that both signals are equal.

## 4.6 Measuring the DAC

The DAC characteristics are measured by connecting the output of the DAC to the microphone input of a PC. The MCU is then programmed by putting a 24 bit resolution sine on the ROM. For this test the ADC and UART are still enabled to ensure that the results match the results of the total system as close as possible. Next Matlab is used to measure the audio input for 10 seconds with a sampling frequency of 32 kHz and resolution of 24 bits. Finally an FFT with a Kaiser window is applied to the measured signals and the THD and SNR are calculated and compared with the requirements.

### 4.6.1 Results

Below the results of the measurements for sines at 20 Hz, 100 Hz, 500 Hz and 1000 Hz are shown at an output voltage of 1.4V Pk-Pk. The measurements itself are included in appendix B. The logarithmic scales are translated so that the signal level is at 0 dB to increase readability.

**THD** Requirement 2.2.3. requires the THD to be at most 5% or -26 dB of the THD mask in appendix A. This required maximum THD and the measured THD are shown in table 8. It is clear that the DAC meets the specs.

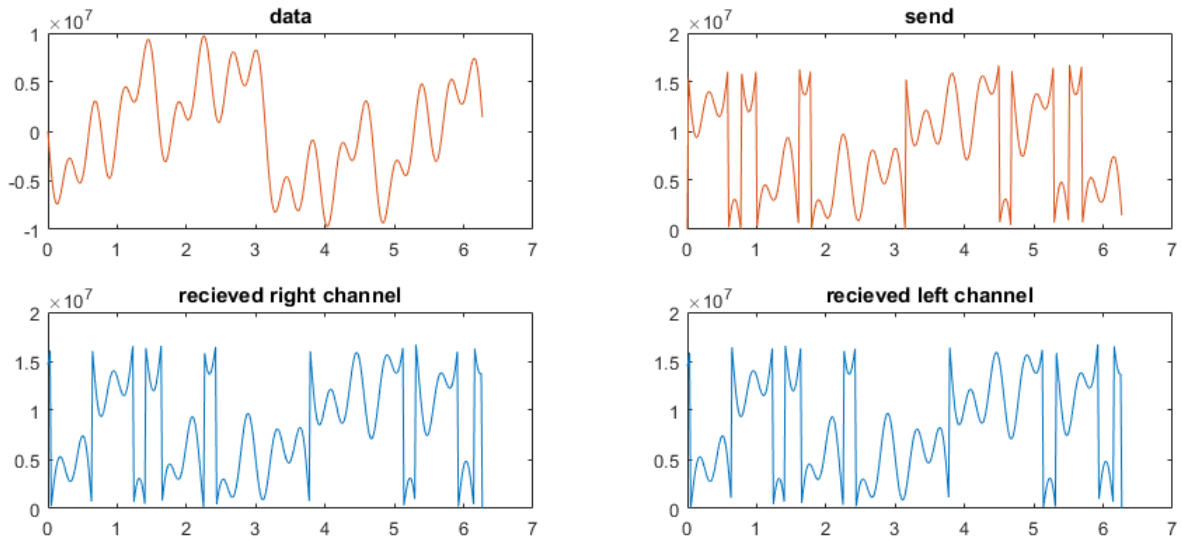



**Figure 32:** Measurement of the UART logging. The two top signals are the reference signals uploaded to the ROM of the MCU and the bottom two the signals received signals from the UART module of the MCU.

**Table 8:** Results of the THD of the DAC

| Frequency | Required | 1st Harmonic | 2nd Harmonic | 3rd Harmonic |
|---|---|---|---|---|
| 20 Hz | -1.31 dB | -43.31 dB | -52.46 dB | -57.98 dB |
| 100 Hz | -32.04 dB | -68.93 dB | -73.22 dB | -78.64 dB |
| 500 Hz | -32.04 dB | -81.70 dB | -85.64 dB | -89.57 dB |
| 1000 Hz | -32.04 dB | -85.34 dB | -90.97 dB | -92.80 dB |

**SNR** The noise is smaller than the side lobes of the window at every measurement. These side lobes have a amplitude smaller than -100 dB which means that the SNR is higher than 100 dB. This is the requirement by specification 2.2.4.

# 5 Conclusion

This thesis was set to implement the digital part of a motional feedback system. There were four parts to this:

- Converting an analog signal to a digital signal.
- Having the possibility to process this digital signal using custom C code.
- Having a logging option, to output the digital signal to an external PC.
- Converting the processed digital signal back into an analog signal.

**ADC** From the design it follows that the conversion from an analog signal to a digital signal has to be implemented using a 24 bits Sigma Delta ADC that outputs its signal using the standard Philips $I^2S$ protocol. As ADC the PCM1808 is chosen. The implementation of this part however is not finished before the deadline of this thesis.

**Microcontroller** The possibility to process the digital signal using C code is implemented by using a microcontroller as processing unit and providing the digital input signal in a buffer. The processed output can then be written to another output buffer.

**Logging** The logging option is implemented by a UART USB connection to a PC which logs its data at every interrupt generated by the ADC. This ensures that every data that is read into the microcontroller is logged to the PC. The baud rate of the UART is chosen to be as high as possible to reduce the time spent logging.

**DAC** The conversion from a digital signal to an analog signal is implemented by the CS43L22. This is a 24 bits Delta Sigma DAC that reads its data using the standard Philips $I^2S$ protocol. The CS43L22 also has a class D audio amplifier that is configurable using the $I^2C$ protocol. The results from the measurements meet the requirements.

# 6 Reflection and future work

Reflecting on our project, we are pleased by the implementation and results obtained so far. We learnt a lot from the different communication protocols and how they are implemented on MCUs, ADCs and DACs.

The results obtained with the DAC meet the requirements with a lot of room to spare for the ADC. We would have wished that the ADC had given usable results, but the preliminary results were insufficient to include in this thesis. However we can conclude that if the ADC operates as specified in the data sheets, the THD and SNR would meet the specifications.

For further research the first step would be to implement the same setup with a working ADC, possibly a different one. Another possibility is to implement the processing unit with an FPGA, to increase the throughput of the system since the DAC, ADC and IIR filter can be processed in parallel. The most important reason that this was not possible in our research was that another subgroup had to be able to implement the controller in C.

# A THD Mask

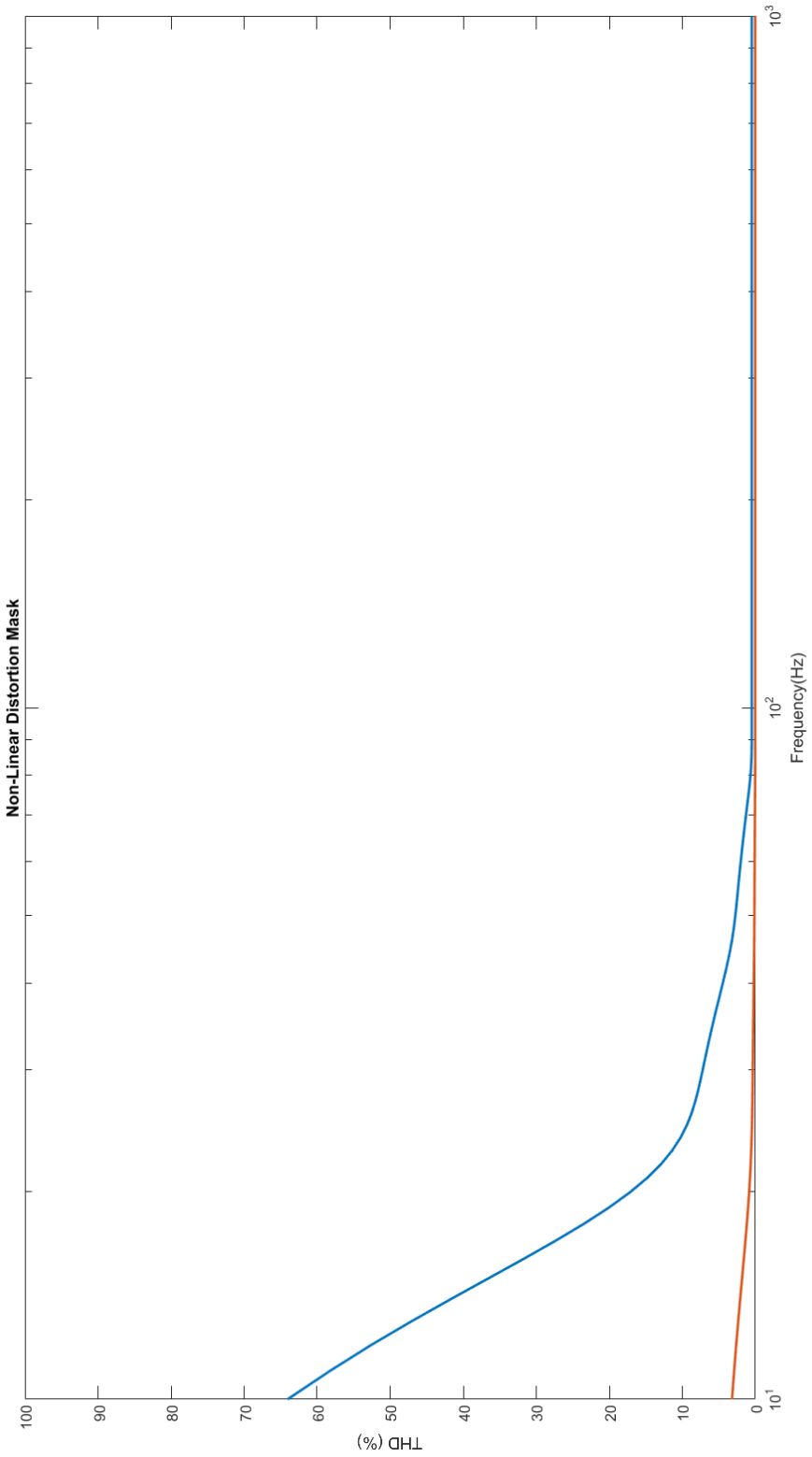



**Figure 33:** The Mask used for the THD specification

# B DAC measurements

## B.1 20 Hz

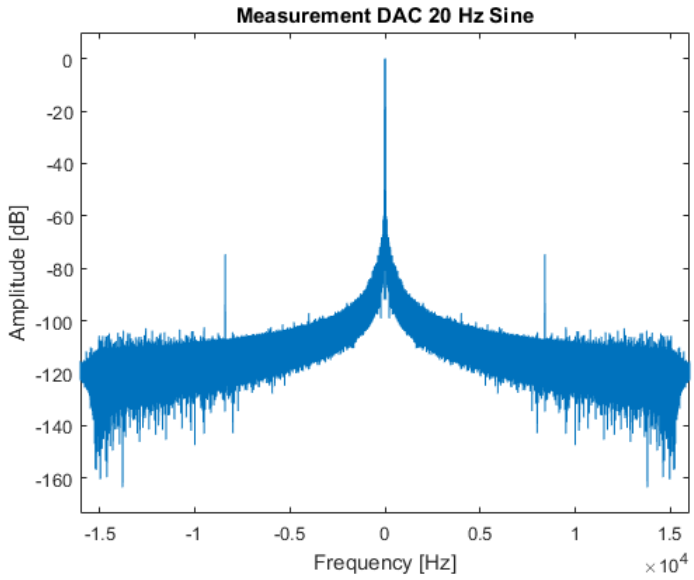



**Figure 34:** Measurement of the DAC at 1.4V Pk-Pk with a sine wave of 20 Hz at a sampling frequency of 32 kHz.

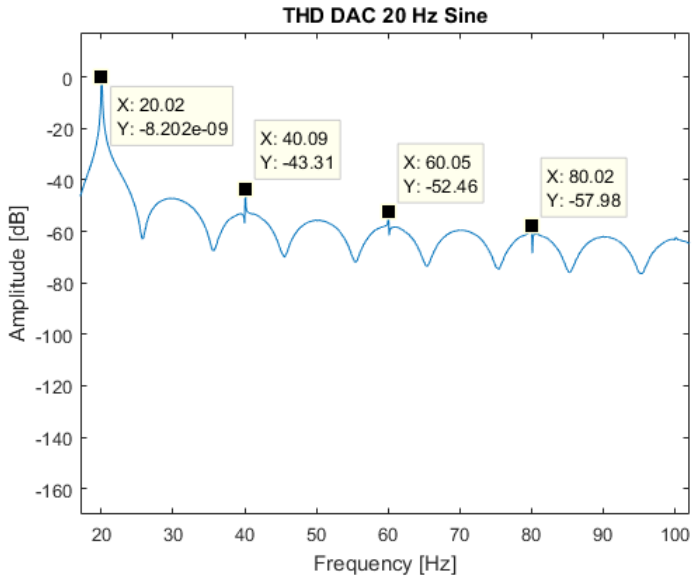



**Figure 35:** Measurement of the DAC zoomed in at the three to measure the THD.

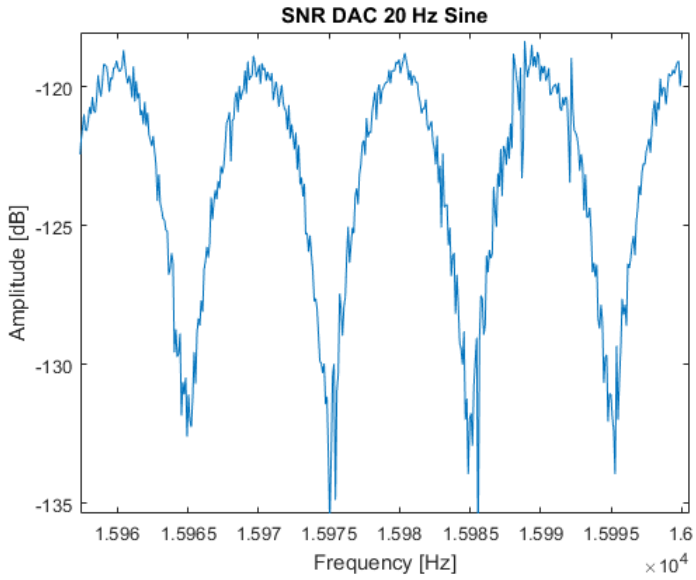



**Figure 36:** Measurement of the DAC zoomed in at the highest frequency for lowest side lobe amplitude to see the noise.

## B.2 100 Hz

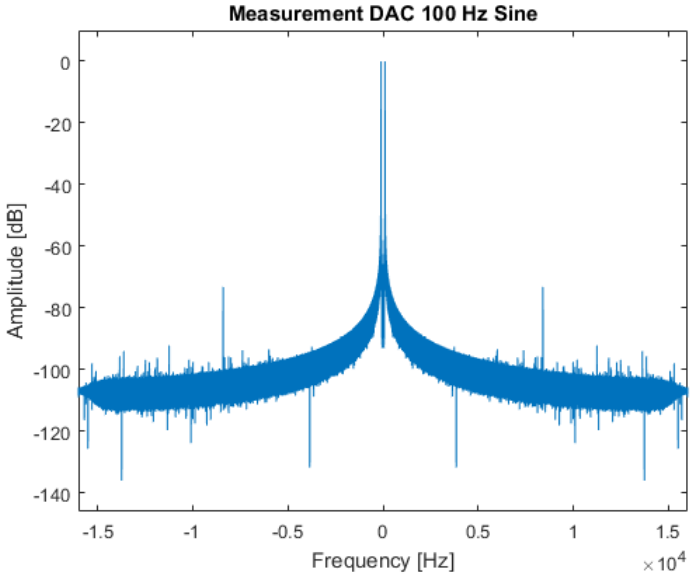



**Figure 37:** Measurement of the DAC at 1.4V Pk-Pk with a sine wave of 100 Hz at a sampling frequency of 32 kHz.

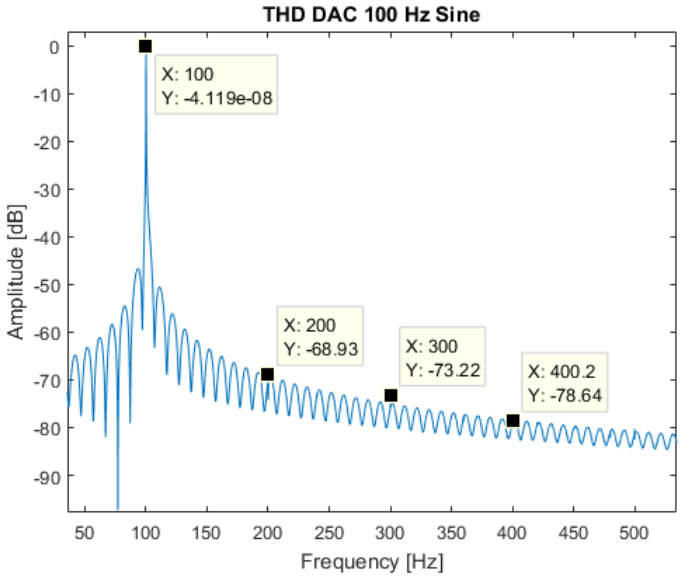



**Figure 38:** Measurement of the DAC zoomed in at the three to measure the THD.

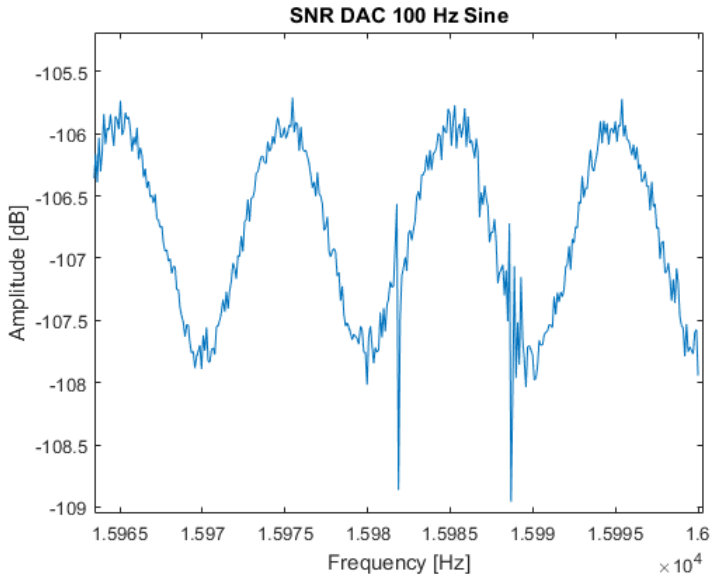



**Figure 39:** Measurement of the DAC zoomed in at the highest frequency for lowest side lobe amplitude to see the noise.

## B.3 500 Hz

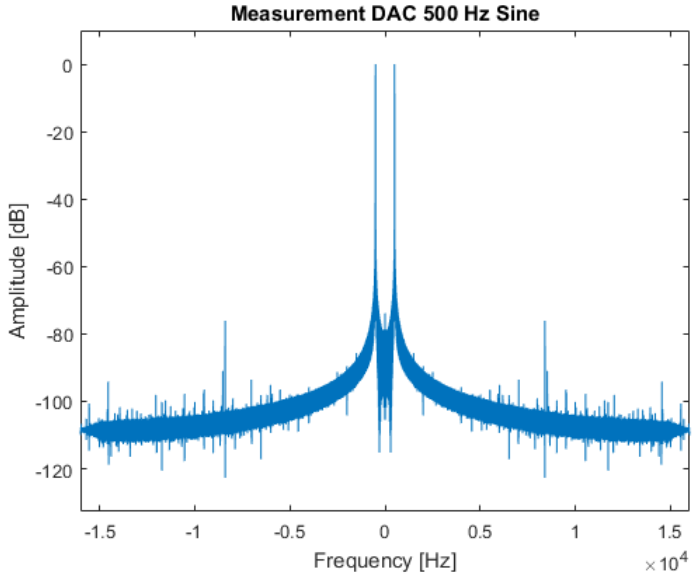



**Figure 40:** Measurement of the DAC at 1.4V Pk-Pk with a sine wave of 500 Hz at a sampling frequency of 32 kHz.

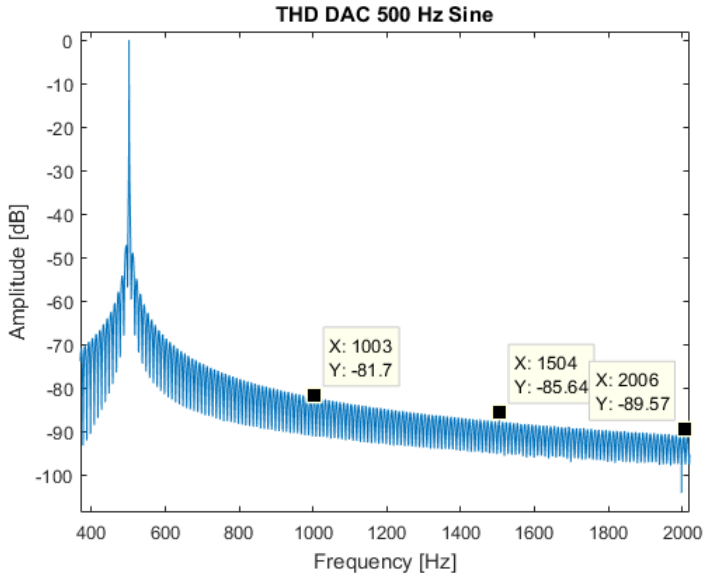



**Figure 41:** Measurement of the DAC zoomed in at the three to measure the THD.

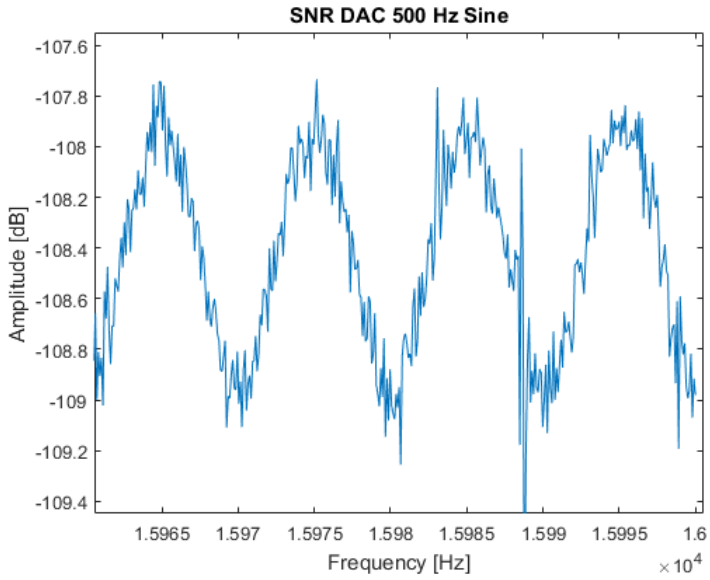



**Figure 42:** Measurement of the DAC zoomed in at the highest frequency for lowest side lobe amplitude to see the noise.

## B.4 1000 Hz

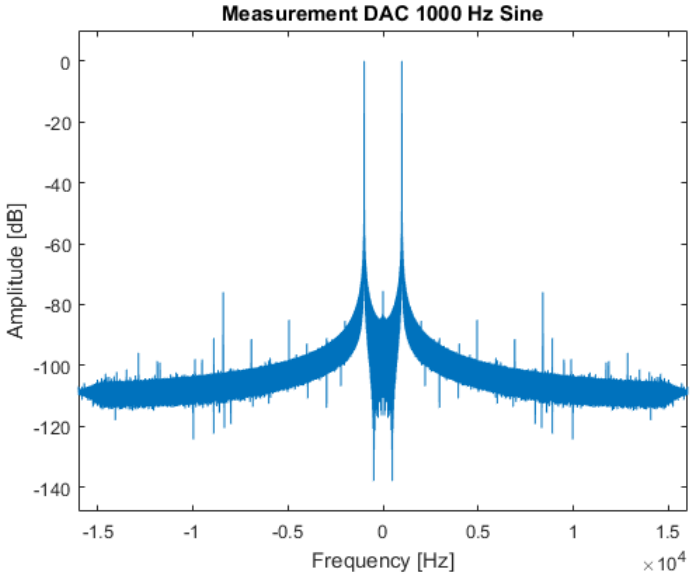



**Figure 43:** Measurement of the DAC at 1.4V Pk-Pk with a sine wave of 1000 Hz at a sampling frequency of 32 kHz.

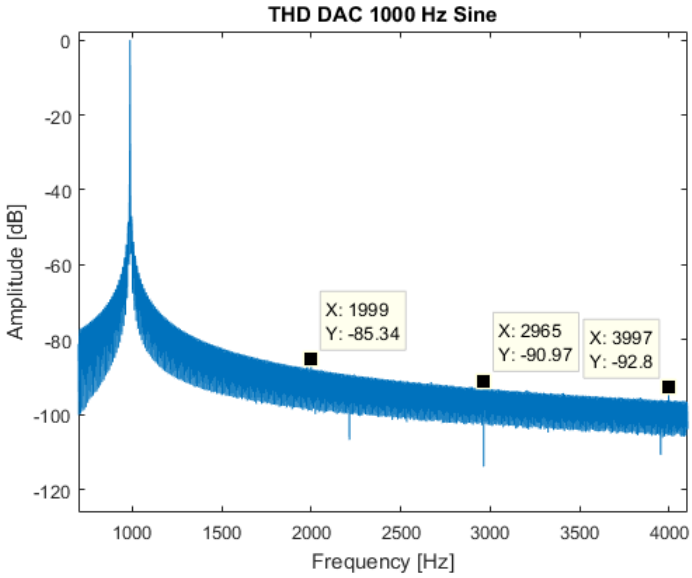



**Figure 44:** Measurement of the DAC zoomed in at the three to measure the THD.

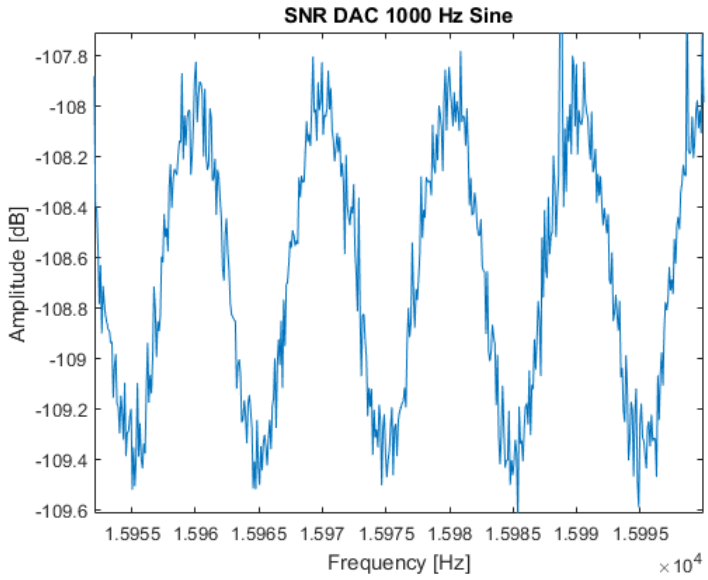



**Figure 45:** Measurement of the DAC zoomed in at the highest frequency for lowest side lobe amplitude to see the noise.

# References


[1] J. A. Klaassen, S. H. de Koning *Bewegingstegenkoppeling bij luidsprekers*, Philips Technisch Tijdschrift, jaargang 29, no. 5, pp. 152-161, 1969

[2] Jaco Salentijn, Oscar de Groot, *Amplifier Design in a Motional Feedback Audio System*, Department of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, Netherlands, 2016

[3] Wouther Bons, Tijs Hol *Digital Control in a Motional Feedback Audio System*, Department of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, Netherlands, 2016

[4] M. van der Veen *De DSS-930 uitgekleed...*, Home Studio, jaargang 9, nr. 9, pp. 14-17, 1992

[5] R. Valk, *Control of Voicecoil transducers,* M.S. thesis, 3mE, Delft Univ. of Technology, Delft, 2013

[6] I2C-Bus. *I2C* [Online]. Available: http://www.i2c-bus.org/

[7] Phillips Semiconductors. *I2S* [Online]. Available: https://www.sparkfun.com/datasheets/BreakoutBoards/I2SBUS.pdf

[8] Motorola. (2003, February). *SPI Block Guide V03.06* [Online]. Available: https://web.archive.org/web/20150413003534/http://www.ee.nmt.edu/~teare/ee308l/datasheets/S12SPIV3.pdf

[9] Texas Instruments. (2010, November). *Universal Asynchronous Receiver/Transmitter (UART)* [Online]. Available: http://www.ti.com.cn/cn/lit/ug/sprugp1/sprugp1.pdf

[10] ST. (2016, February). *USER MANUAL Discovery kit with STM32F407VG MCU* [Online]. Available: http://www.st.com/content/ccc/resource/technical/document/user_manual/70/fe/4a/3f/e7/e1/4f/7d/DM00039084.pdf/files/DM00039084.pdf/jcr:content/translations/en.DM00039084.pdf

[11] ARM. (2010, March, 2). *Cortex-M4 Technical Reference Manual* [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.ddi0439b/DDI0439B_cortex_m4_r0p0_trm.pdf

[12] ST. (2015, September). *Description of STM32F4xx HAL drivers* [Online]. Available: http://www.st.com/content/ccc/resource/technical/document/user_manual/2f/71/ba/b8/75/54/47/cf/DM00105879.pdf/files/DM00105879.pdf/jcr:content/translations/en.DM00105879.pdf

[13] ST. (2013, June). *STM32F405xx, STM32F407xx Data Sheet* [Online]. Available: http://www.st.com/content/ccc/resource/technical/document/datasheet/ef/92/76/6d/bb/c2/4f/f7/DM00037051.pdf/files/DM00037051.pdf/jcr:content/translations/en.DM00037051.pdf

[14] Cirrus Logic. (2010, March). *Low Power, Stereo DAC w/Headphone & Speaker Amps* [Online]. Available: https://www.cirrus.com/cn/pubs/proDatasheet/CS43L22_F2.pdf

[15] TI. (2006, April). *PCM1808 Single-Ended, Analog-Input 24-Bit, 96-kHz Stereo ADC* [Online]. Available: http://www.ti.com/lit/ds/symlink/pcm1808.pdf