VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Tran Minh Hien

Environmental noise level monitoring device

Thesis report accepted 08.05.2020
Thesis report updated 01.06.2020

Information Technology
2020

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Bachelor of Engineering

ABSTRACT

Author	Tran Minh Hien
Title	Environmental noise level monitoring device
Year	2020
Language	English
Pages	36 + 0 Appendices
Name of Supervisor	Jani Ahvonen

This thesis project explored the possibility of a comprehensive sound level monitoring device on recently developed components, in collaboration with TJK Tietolaite Oy to ensure the real-life practicality of the end product. The device's goal was to monitor sound level continuously and communicate wirelessly over long distances.

The project examined the requirement of the IEC 61672-1:2013 and based on these requirements, build a proof-of-concept of the device, using MEMS microphone. STM32 Nucleo-H742ZI development board and the CMSIS DSP library.

The project was able to build a functional prototype that could give sound level measurment. However, its performance was not adequtely asserted and finely calibrated due to a lack of equipment as a result of the current Covid-19 pandemic.

Keywords Sound level meter, digital processing

CONTENTS

LIST OF FIGURES AND TABLES

1 INTRODUCTION

In spite of its harmful impact on the activity of human and animal lives, environmental noise is often inadequately monitored and documented. One reason for this is perhaps the lack of a low-cost real-time sound level measuring system, as current commercial equipment is too expensive and operate manually. The goal of this thesis is to explore the possibility of a real-time sound level monitoring system that is low-cost, low-maintenance and capable of working remotely over long distances.

2 REQUIREMENTS

2.1 IEC 61672-1:2013 standard

2.1.1 General

IEC 61672-1:2013 (Electroacoustics - Sound level meters - Part 1: Specifications) is a standard issued by the International Electrotechnical Commission specifying performance requirements for sound level measuring devices that aims to measure sound in the range of human being.

2.1.2 Types of device

The standard gives specification for three types of sound level measuring devices:

- A time-weighting sound level meter measuring frequency-weighted sound level
- An integrating sound level meter measuring time-averaged, frequency-weighted sound level
- An integrating sound level meter measuring frequency weighted sound exposure level

If the sound level meter only indicates sound exposure level, time-average sound level can be determined by the following equation:

$$L_{eq,T} = 10log\left(\frac{E_T}{p_0^2 T}\right) \quad (1)$$

With:

- $L_{eq,T}$: time-average sound level during T (db)
- E_T: sound exposure during T $(Pa^2 s)$
- T: measurement time interval (s)
- p_0: sound pressure level reference (Pa). Value: $p_0 = 20 \times 10^{-6} Pa$

2.1.3 Performance class

Two performance classes are specified for sound measuring, class 1 and class 2. Class 1 specification is generally more demanding than class 2 specification. Therefore, class 2 devices may provide some class 1 capabilities, but to be class 1 devices they must satisfy

all relevant class 1 requirements. A device, however, can be specified as class 1 in one configuration, and as class 2 in a different configuration (such as a different microphone, windscreen, etc.)

Requirement	Class 1	Class 2
Required frequency weighting	A, C	A
Acceptance limit passband $(-3dB)$	$20Hz - 10kHz$	$40Hz - 3.15kHz$
Acceptance limit ripple	1dB	1.5dB
Acceptance limit at reference frequency $(1kHz)$	0.7dB	1dB
Absolute linear deviation limit	0.8dB	1.1dB
Relative linear deviation limit	0.3dB	0.5dB

Table 1: Comparison between class 1 and class 2 requirement

2.1.4 Testing condition

The frequency responses specified for the standard apply to sound incidents in an acoustic free field from one principal direction or successive incidents from random directions.

Specifications in the standard apply under the reference environmental conditions:

- Air temperature: 23℃
- Static air pressure: 1 atm
- Relative humidity: 50%

2.1.5 General requirements

The standard defines a general sound level meter to have a microphone, a preamplifier, a signal processor, and a display device. The standard may be applied to any design of microphone or preamplifier.

The signal processor should perform the following function:

- Amplify the signal with a specified and controlled frequency response
- Form the square of the amplified signal
- Perform time integrating/averaging

The display device should provide a physical display, or storage for measured results that can be view with the device, or a different assisting device or software.

2.1.6 Radio-frequency interference

To specify the radio-frequency emission and immunity of the sound level meters, they are classified into three group:

- Group X: self-contained instruments with an internal battery that requires no external connection for normal operation
- Group Y: self-contained instruments powered by external sources that require no other connection for normal operation
- Group Z: instruments that require more than one piece of equipment for normal operation. The equipment must be connected in some way for the instrument to function, and each part may be powered either internally or externally

2.1.7 Frequency weighting

Three methods of frequency weighting are provided by the standard: A, C and Z. All frequency weighting methods are designed to have a 0dB gain at 1kHz. The frequency weighting and acceptance limit shall apply for either the free-field response or random-incident response.

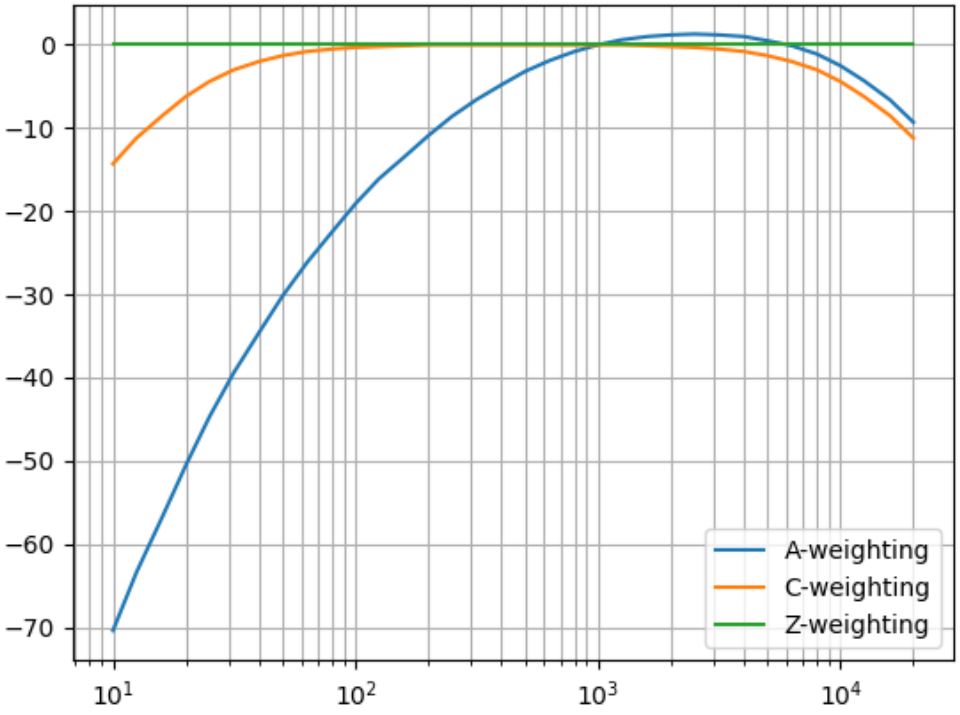

Figure 1: Frequency response of A, C and Z-weighting

Frequency-weighting A is required for both class 1 and class 2 devices. Frequency-weighting C is required for class 1 devices. Frequency-weighting Z is optional.

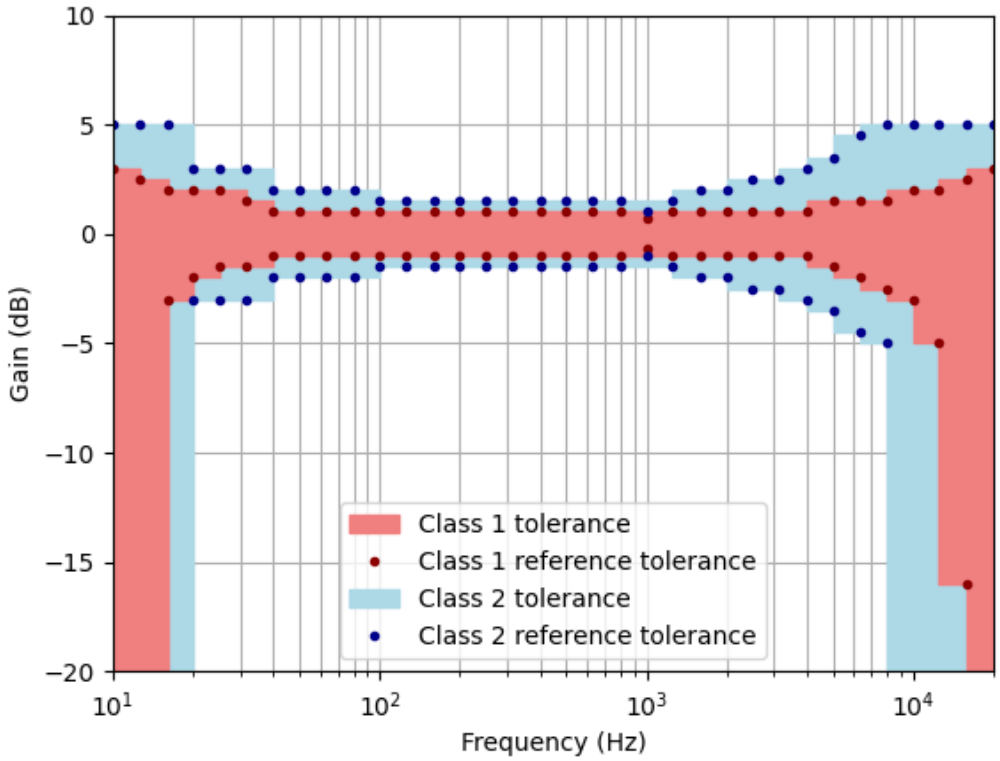

Figure 2: Acceptance limit for performance class 1 and 2

The acceptance limit dictates the maximum variation from the frequency response of the frequency-weighted signal to the targeted frequency-weighting design goal. This limit tolerates a passband filter. The tolerance at the nominal reference frequencies is given in the standard, while the tolerance for frequency between these references is the larger of the tolerances of its nearest reference frequency.

2.1.8 Time weighting

The time weighting filter serves the purpose of damping the sudden changes in the environment to a smoother signal display for the human operator. Two time-weighting methods are specified in the standard, fast mode (F) and slow mode (S). Time frequency F is required for the time-weighting sound level meter.

Design goal	Fast mode (F)	Slow mode (S)
Time constant (ms)	125	1000
Decay rate (dB/s)	34.7	4.3
Decay rate, acceptance limits (dB/s)	-3.7; +3.8	-0.7; +0.8

Table 2: Design goals for time-weighting method F and S

The time-weighting process is squaring the frequency-weighted signal, applying a low-pass filter with one real pole at the inverse of the time constant, taking the base-10 logarithm, then displaying or averaging/integrating this result.

2.1.9 Level linearity

The measured signal level should be a linear function of the sound pressure across the whole dynamic range. Acceptance limit on level linearity apply for electrical signal injected into the preamplifier of the input device. This requirement has been updated from the 2003 version of the standard in order to accommodate for the use of MEMS microphone, as that version of the standard required the electrical signal to be injected through the microphone, which is practically impossible for the MEMS microphone considering their size, and that they often have embedded circuit inside of the microphone.

A reference sound pressure level should be chosen and specified in the Instruction Manual, and the expected level for each frequency should be the sum of this reference level and the gain of the frequency weighting at that frequency. The extent of the linear operating range should be at least 60dB at 1kHz.

Level linearity deviation shall not exceed 0.8dB for class 1 and 1.1dB for class 2. Any 1dB to 10dB change in the input level shall cause the same change in the output signal, with a tolerate of 0.3dB for class 1 and 0.5dB for class 2. It should be noted that time-weighting F, if applicable, may cause ripple at low frequencies.

At 1kHz, linear operating ranges shall overlap by at least 30dB for time-weighting measuring, or 40dB for time-averaging/integrating measuring.

The nominal operating range, in which the device can measure without displaying of under-range or overload condition, shall be specified for each available frequency weighting in the Instruction Manual.

2.1.10 Level range

Sound level meters may have more than one level range. The range shall be specified by the lower and upper limits of the nominal A-weighted sound level at 1kHz

2.1.11 Start-up time

Specification of the standard may be applied only after a time period has passed since powering up the device. The device may reach equilibrium with the ambient environment during this start-up time. The maximum start-up time for the device is two minutes.

2.2 Technical requirements

2.2.1 Sound level measurement

The requirements for the sound level measurement components are the requirements for a class 1 device according to IEC-61672 for sound level meter, in addition to requirements from TJK Tietolaite Oy based on their needs:

- Enviromental endurance
- Low power consumption
- The filtering process done mainly by the software

2.2.2 Wireless communication

The requirements for the measurement system are:

- The system will consist of a number of remote sensor node sending data back to one central server.
- The wireless technology used should be able to cover a large physical area with a small number of nodes, either via communication over long distance (up to 1km), or capable of time synchronized mesh networking.

2.2.3 Performance requirement

The sensor nodes will operate remotely outdoor for at least one month without manual maintenance. The nodes will be powered by battery, and must endure the harsh Finnish weather, as well as other unforeseen environmental conditions.

3 SYSTEM OVERVIEW

3.1 Overall

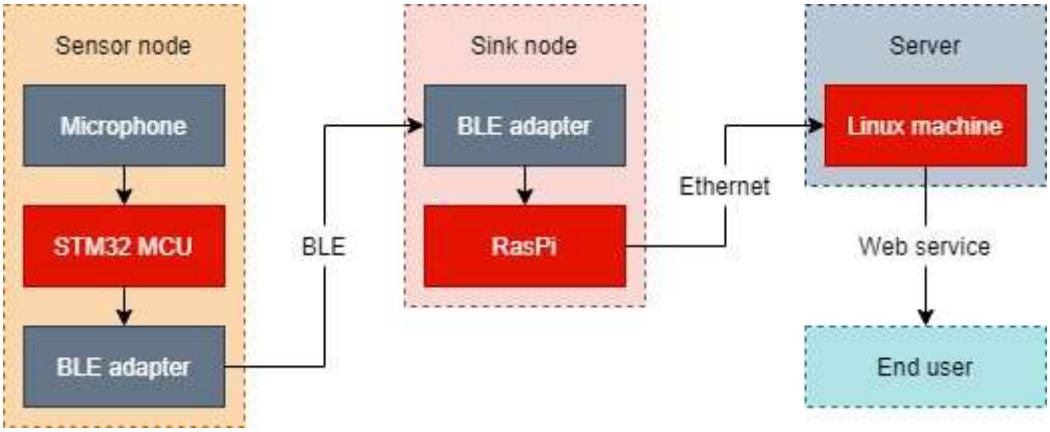

Figure 3: Overall design of the project

3.2 Sensor node

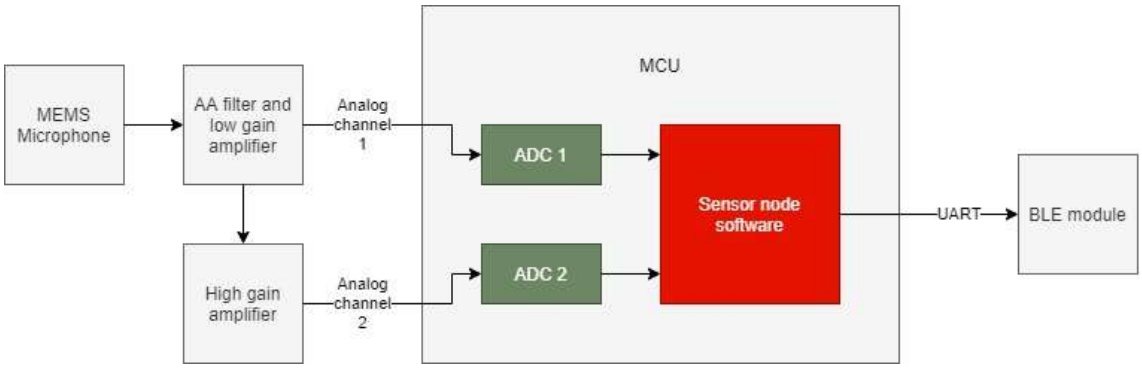

Figure 4: Block diagram of the sensor node hardware

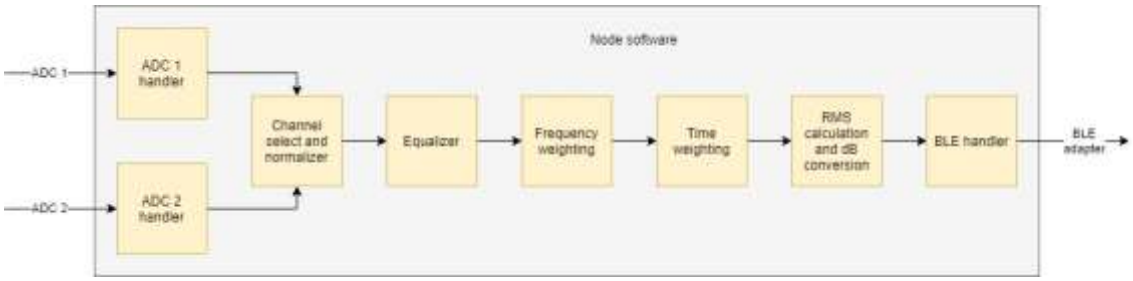

Figure 5: Block diagram of the sensor node hardware

4 DESIGN AND TECHNOLOGY PRESTUDY

4.1 MEMS microphone

4.1.1 MEMS technology

MEMS, or microelectromechanical systems, are tiny intelligent objects with the capabilities of performing electrical or mechanical functionality. Their dimensional size may vary from centimeters down to micrometers, and can be manufactured in parallel by using different technology already well mastered in microelectronics /2/. As a result, MEMS technology creates the possibility of small-sized, low-cost, low-power components with mechanical capabilities, perfectly suited for small and portable devices.

4.1.2 MEMS microphone

MEMS microphones are characterized by their small size, low power consumption, low cost, high sensitivity and flat frequency response, and are suitable for a sound level meter, although they generate more noise than their earlier counterpart, and are vulnerable to physical elements (shock and vibration, air temperature and moisture).

While MEMS microphones may use piezoelectric, piezo-resistive, optical or capacitive principles to convert sound pressure into electrical signal /11/. 80% of currently produced MEMS microphones are capacitive transductors, with the advantage of high sensitivity, low power consumption, and better conformity to batch production /6/.

The main operational principal of MEMS microphone is similar to that of electret condenser microphone. The transducers consist of two conductive plates separated by air in a chamber, with one plate having a layer of permanently polarized dielectric material, which create charges on the plates. One of the plates is stationary (the back plate), while the other plate can vibrate under air pressure changes caused by sound waves (the diaphragm). This vibration changes the capacity of the transducers, and these changes can be converted into analog signal by different types of interface circuit. The chamber enclosed the transducer has an open acoustic port hole for the sound to come in, and depend on the position of the port, a microphone is categorized as top port or bottom port. Bottom port microphone has its back plate perforated in order for the sound to penetrate. The volume between the back of the diaphragm and the wall of the chamber is call the back

volume, and this volume can majorly affect the acoustic performance of the microphone. /11/

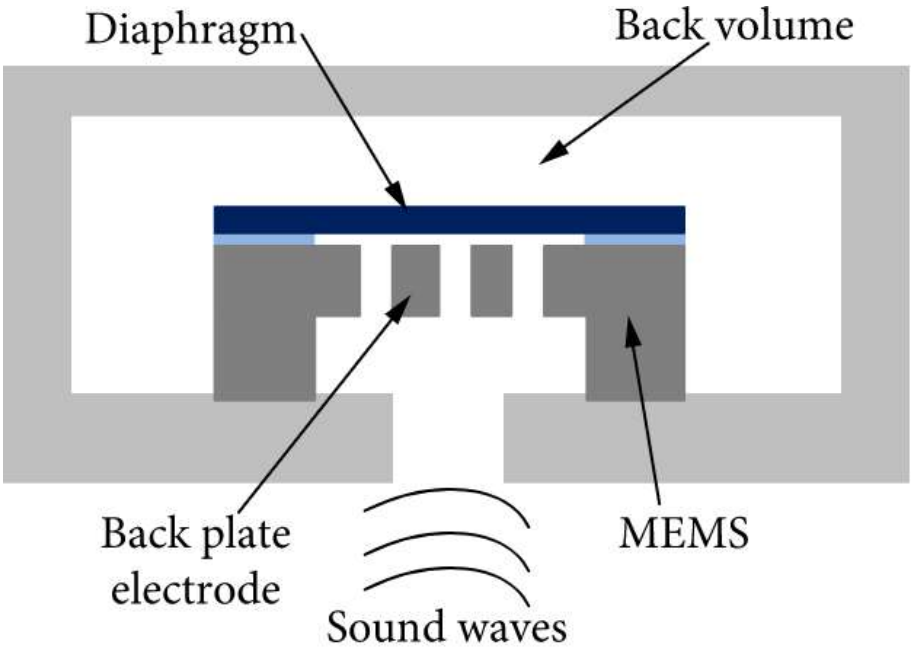

Figure 6: Figure: cross section of a bottom port MEMS microphone /11/

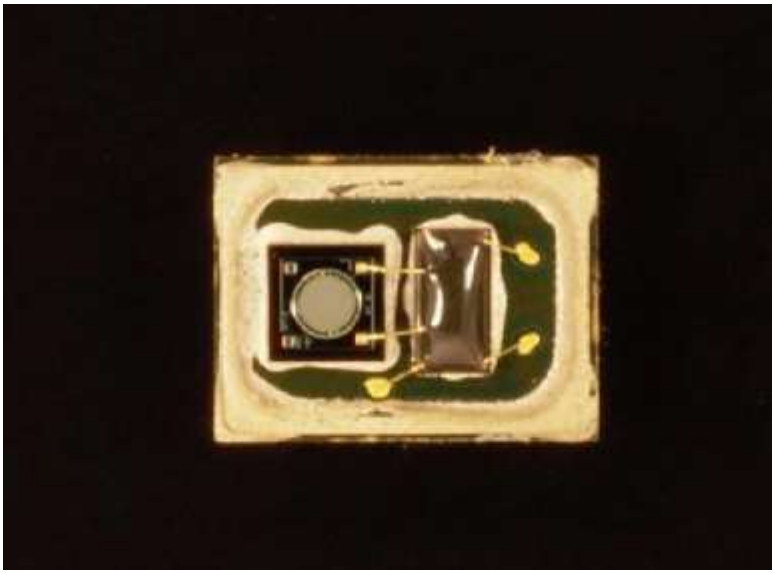

Figure 7: Transducer (left) and ASIC (right) of an analog MEMS microphone /5/

A typical capacitive MEMS microphone package is consisted of two components: the MEMS transducer and the application-specific integrated circuit (ASIC) for the microphone. The purpose of the ASIC is to convert the changes in the capacitive between the

plates of the transducer into analog signal, and to bring the output impedance of the microphone down to a more usable value /5/, still the output impendence of MEMS microphones remains relatively high.

The frequency responses of capacitive MEMS microphones are comparable to the best electret condenser microphone on the market, as their smaller sizes increase the MEMS microphone resonance frequency, and consequently, extend the region of flat response before resonance. However, most MEMS microphones struggle to archive a SNR higher than 60dB. /10/

MEMS microphones, while having low environmental endurance, are less vulnerable to temperature and vibration than their traditional electret condenser microphone, and can be reflow soldered. /11/

4.1.3 ICS-40300

ICS-40300 is a model of analog MEMS microphone produced by InvenSense. This model was chosen based on various benefits that it provides: low noise, flat frequency response, low power consumption, and most importantly, high maximum sound level pressure limit. The ICS-40300 microphone has a wide flat response zone from 6Hz up to 20kHz, a dynamic range of 99dB and a high acoustic overload point of 130dB.

Important characteristics of this model can be viewed in the following table:

Parameter	Value
Directionality	Omnidirectional
Sensitivity (dBV)	-45
Signal-to-noise ratio (dB)	63
Equivalent input noise (upper SPL limit, dB)	31
Acoustic overload point (lower SPL limit, dB)	130
Supply voltage (V)	1.5 – 3.6
Output Impedance (Ω)	200
Output DC offset (V)	0.8
Maximum output voltage (V_{rms})	0.355
Temperature range (℃)	-40 – +85

Table 3: Typical characteristic of ICS-40300 microphone

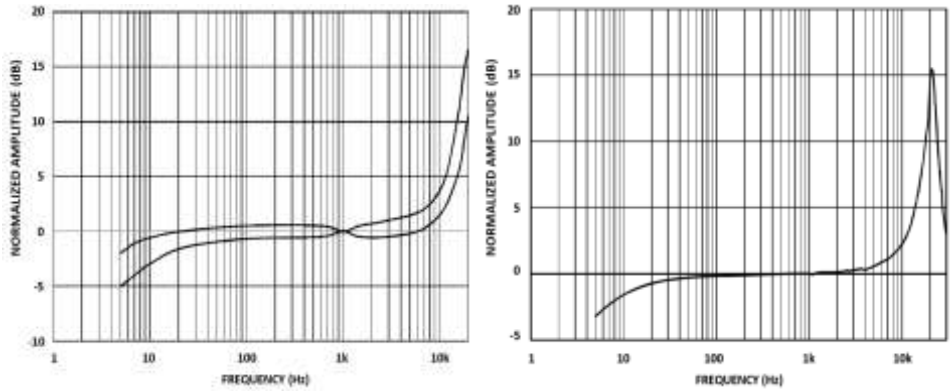

Figure 8: Frequency respond mask (left) and typical frequency respond (right) of the ICS-40300

The ICS-40300 microphone has a typical frequency of a MEMS microphone with low gain at low frequency, and has a peak of about $15dB$ near $20kHz$. The typical sensitivity may vary from $-47dBV$ to $-43dBV$, but it is typically $-45dBV$.

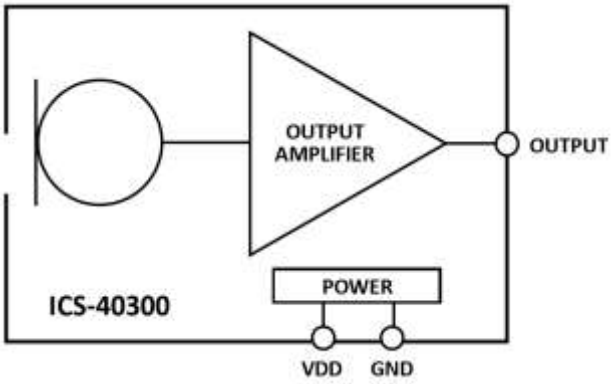

Figure 9: ICS-40300 block diagram

Like many typical MEMS microphones, the ICS-40300 microphone has a built-in amplifier inside of its package. The microphone has 6 pins: 4 ground pins, 1 power supply pin and 1 analog output pin.

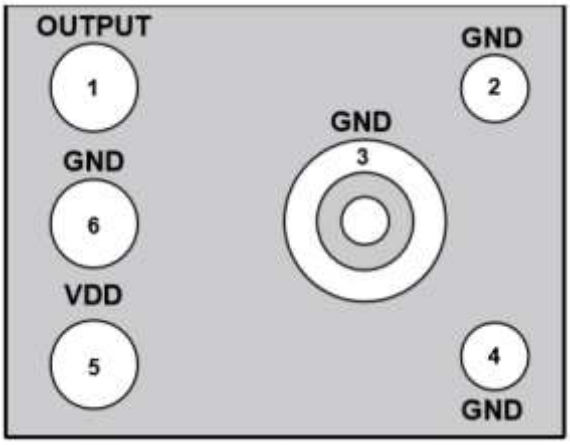

Figure 10: Pin configuration of ICS-40300 microphone (bottom view)

4.2 Operational Amplifier

For this project, the OPA2340 model of operational amplifier was chosen dues to many requirements:

- Single-sided supply: The OpAmp will only be supply with one polarity and no virtual ground
- Rail-to-rail output: Output signal close to the supply voltage limit will reduce clipping for the high gain amplifier to better utilize its dynamic range
- Low gain error: Lower gain error will reduce the ADC error margin

4.3 Microcontroller

For this project, the STM32H743ZI was chosen due to its functionalities:

- 32-bit ARM Cortex-M7 core with double precision FPU and DSP instructions, up to 480*MHz* clock frequency
- 1 MB of RAM
- 3 16-bit ADC, with 36 channels and up to 3.6 MSPS

4.4 Frequency weighting

The characteristic of the A-weighting filter is defined in /4/:

$$A(f) = 10log\left(\frac{F_4^2 f^4}{(f^2 + F_1^2)\sqrt{f^2 + F_2^2}\sqrt{f^2 + F_3^2}(f^2 + F_4^2)}\right) - A_{1000} \quad (2)$$

with: A_{1000}: gain offset to normalize total gain to $0dB$ at $1kHz$ (dB)

F_1, F_2, F_3, F_4: filter poles designed base on $F_A = 10^{2.45}Hz$

$F_1 = 20.60\ Hz$

$F_2 = 107.7\ Hz$

$F_3 = 737.9\ Hz$

$F_4 = 12194\ Hz$

This calls for a filter with the frequency response of:

$$G = \frac{F_4^2 f^4}{(f^2 + F_1^2)\sqrt{f^2 + F_2^2}\sqrt{f^2 + F_3^2}(f^2 + F_4^2)} \quad (3)$$

With $\omega = 2\pi f \Rightarrow f = \frac{\omega}{2\pi}$, (3) becomes:

$$G = \frac{F_4^2 \left(\frac{\omega}{2\pi}\right)^4}{\left[\left(\frac{\omega}{2\pi}\right)^2 + F_1^2\right]\sqrt{\left(\frac{\omega}{2\pi}\right)^2 + F_2^2}\sqrt{\left(\frac{\omega}{2\pi}\right)^2 + F_3^2}\left[\left(\frac{\omega}{2\pi}\right)^2 + F_4^2\right]} \quad (4)$$

$$= \frac{(2\pi F_4)^2 \omega^4}{(\omega^2 + (2\pi F_1)^2)\sqrt{\omega^2 + (2\pi F_2)^2}\sqrt{\omega^2 + (2\pi F_3)^2}(\omega^2 + (2\pi F_4)^2)} \quad (5)$$

$$= \frac{|(2\pi F_4)^2 (j\omega)^4|}{|j\omega + 2\pi F_1|^2 |j\omega + 2\pi F_2||j\omega + 2\pi F_3||j\omega + 2\pi F_4|^2} \quad (6)$$

From (6), and that $G(\omega) = |H(j\omega)|$, we can choose the following transfer function to produce the desired frequency response:

$$H(s) = \frac{4\pi^2 F_4^2 s^4}{(s + 2\pi F_1)^2 (s + 2\pi F_2)(s + 2\pi F_3)(s + 2\pi F_4)^2} \quad (7)$$

Based on /9/, matched-z transform method yields the best result for designing the A-weighting digital filter at $f_s = 48\ kHz$. Apply matched-z transform to (7) at sample time $T = f_s^{-1} = 48000^{-1} = 208.33\mu s$ by substituting $z = e^{sT}$ to acquire the filter in the z-plane (the gain for the filter is ignored and will be compensated at the end of the filter process):

$$H(z) = \\ = \frac{(1-e^{0}z^{-1})^{4}}{(1-e^{-2\pi F_1 T}z^{-1})^{2}(1-e^{-2\pi F_2 T}z^{-1})(1-e^{-2\pi F_3 T}z^{-1})(1-e^{-2\pi F_4 T}z^{-1})^{2}} \\ = \frac{(1-z^{-1})^{4}}{(1-0.9973z^{-1})^{2}(1-0.9860z^{-1})(1-0.9079z^{-1})(1-0.2027z^{-1})^{2}} \quad (8)$$

This digital filter deviates from the analog filter design at high frequency, which can be corrected by adding a low pass section /9/. The final digital filter is:

$$H(z) \\ = \frac{(1+0.3z^{-1})(1-z^{-1})^{4}}{(1-0.9973z^{-1})^{2}(1-0.9860z^{-1})(1-0.9079z^{-1})(1-0.2027z^{-1})^{2}} \quad (9)$$

4.5 Digital filter

The digital filter design in (9) can be implemented with an IIR filter. However, TJK Tietolaite Oy also want to explore the possibility of implementing A-weighting with a FIR filter. Using the open source Python library SciPy, the following method was attempted to design an FIR filter:

- *firls*: FIR filter design using least-squres error minimization
- *firwin2*: FIR filter design using window method base on frequencies and corresponding gains
- *remez*: FIR filter design using the Remez exchange algorithm

Using either *firls* or *remez* method resulted in a filter with major ripple and hence, is not a suitable design for this application. Using *firwin2* method yields a better result, with the resulted filter match the design goal at frequency higher than 100*Hz*, yet it fails to attenuate signal below this threshold. Notably, using 8192 taps, the FIR filter frequency response at 10*Hz* is roughly 5*dB* higher than the design goal, with the tolerance limit at this frequency is 3*dB* for class 1 and 5*dB* for class 2 devices.

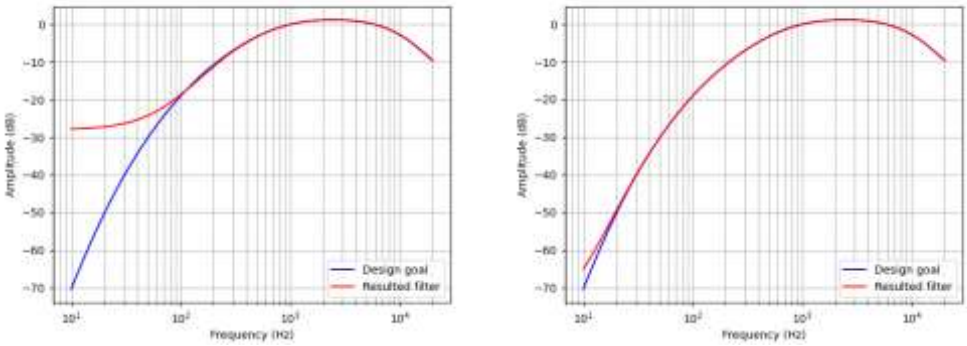

Figure 11: Resulted filter from using the *firwin2* method with 512 taps (left) and 8192 taps (right)

Considering the required complexity for the FIR filter, and the fact that the digital filter in (9) is stable, as all of its poles lie inside the unit circle, the IIR filter is a more reasonable choice of implementation for the A-weighting filter. However, since the poles are in close proximity with the unit circle, instability may occur. To prevent this, the filter can be split into multiple simpler filter using a cascade-form realization with the second order sections. (Risojević, Rozman, Pilipović, Češnovar, & Bulić, 2018)

Using SciPy's function *tf2sos,* the resulted filter is:

$$H(z) = H_1(z)H_2(z)H_3(z) \tag{10}$$

$$H_1(z) = \frac{1 + 0.3000z^{-1}}{1 - 0.4050z^{-1} + 0.0410z^{-2}}$$
$$H_2(z) = \frac{1 - 2.0001z^{-1} + 1.0001z^{-2}}{1 - 1.8957z^{-1} + 0.8970z^{-2}}$$
$$H_3(z) = \frac{1 - 1.9999z^{-1} + 0.9999z^{-2}}{1 - 1.9946z^{-1} + 0.9946z^{-2}}$$

4.6 CMSIS DSP library

4.6.1 CMSIS DSP

Cortex Microcontroller Software Interface Standard (CMSIS) is a hardware abstraction layer for ARM Cortex microcontrollers. This standard provides a high level API for MCU processor and peripherals, RTOS and middleware libraries.

CMSIS Digital Signal Processing library (CMSIS DSP) is a library for digital signal processing supporting both fixed-point (*q7, q15, q31*) and floating-point (*f32*) data types. This library provides implementation for various real and complex number operations, digital filter, matrix calculation and many other DSP related process with SIMD supported on appropriate structure.

4.6.2 CMSIS DSP IIR filter

The CMSIS DSP library provides various implementations for both FIR and IIR filters. For IIR filter, the library supports biquad cascade IIR filter using either direct form I or direct form II structure. These filter are designed to operate in block mode, which means the filter functions will process the input signal block by block.

The filter implementation used in this project is the biquad cascade IIR filter using direct form I structure. This implementation requires the filter to be split into second order bi-quad sections, and these filters will be applied on the signal in series.

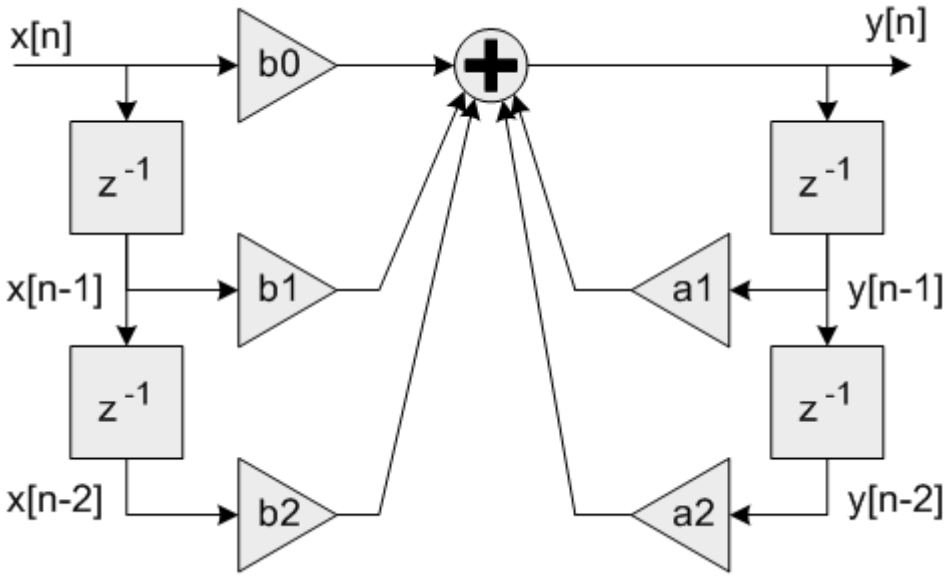

Figure 12: One second order biquad section

The filter is defined by a *arm_biquad_casd_df1_inst_f32* structure, and applied to the signal by the *arm_biquad_cascade_df1_f32* functions. These function is defined in the *arm_math.h* header file.

5 IMPLEMENTATION

5.1 Hardware design

5.1.1 Microphone

The sensitivity of the microphone is calculated by:

$$S_{dB} = 20log\left(S_{V/Pa}\right) = 20log\left(\frac{V}{p}\right) = 20log(V) - 20\log(p) \quad (11)$$

The sound pressure level is calculated by:

$$L_{db} = 20\log\left(\frac{p}{p_0}\right) = 20log(p) - 20\log(p_0) \quad (12)$$

With $p_0 = 20\mu Pa$, let $L_0 = 20log(p_0) = 20log(20 \times 10^{-6}) \approx -94dB$, (3) became:

$$20log(p) = L_{dB} - 94 \quad (13)$$

Combine (2) and (4):

$$\begin{aligned} &20\log(V) = S_{dB} + L_{dB} - 94 \\ &\Rightarrow V = 10^{\frac{S_{dB}+L_{dB}-94}{20}} \\ &\Rightarrow L_{db} = 20\log(V) - S_{dB} + 94 \end{aligned} \quad (14)$$

with: V: peak voltage signal output (V)

S_{dB}: microphone sensitivity (dB)

L_{dB}: sound pressure level (d)

The sensitivity of the ICS-40300 microphone may vary from $-47dB$ to $-43dB$, with the typical sensitivity being $S_{dB,typ} = -45dB$. Therefore, the typical voltage signal peak of the microphone should be:

$$V(L_{dB}) = 10^{\frac{L_{dB}-139}{20}} \quad (15)$$

For all simulation and testing purposes, the reference frequency will be $f = 1kHz$, and the sound level will be $L_{dB} = 94dB$, corresponding to a sound pressure of $1Pa$, as recommended /4/. For $L_{dB} = 94dB$, the typical voltage signal peak will be:

$$V(L_{dB}) = 10^{\frac{L_{dB}-139}{20}} = 10^{\frac{94-139}{20}} = 5.62mV \quad (16)$$

5.1.2 *Analog-to-Digital converter*

The Nucleo-H743ZI evaluation board has 3 built-in 16-bit ADCs, which can provide a dynamic range of:

$$DR = 20log(2^{16}) \approx 96dB. \tag{17}$$

The voltage resolution of the ADC is

$$Q = \frac{V_{ref}}{2^n} = \frac{3.3}{2^{16}} = 50.35\mu V \tag{18}$$

5.1.3 *Anti-alias filter and amplifier*

Since the application needs a dynamic range of $100dB$ ($30dB - 130dB$) yet the ADC only has a maximum dynamic range of $96dB$, the sensor node is designed to have two amplifiers, a low gain amplifier and a high gain amplifier. The two amplifiers are connected to their separated ADC channels.

5.1.4 *Low gain amplifier*

The low gain amplifier is a Sallen-Key filter with integrated gain. Its main function is filtering out high frequency noise that can affect the ADC. In addition to this, the low gain amplifier also brings the microphone signal up to a higher voltage level to better utilize the dynamic range of the ADC.

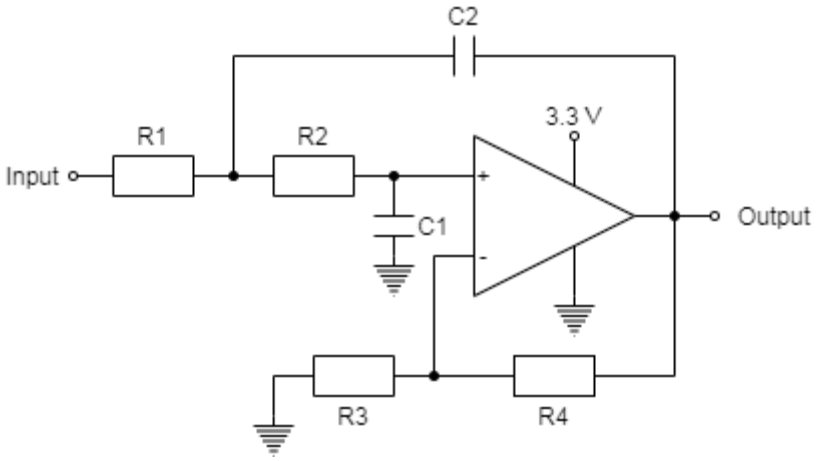

Figure 13: Sallen-Key filter circuit

The maximum peak voltage signal the microphone may produce ($S_{dB,max} = -43dB$, $L_{db,max} = 130dB$) is:

$$V_{max} = 10^{\frac{S_{dB,max}+L_{dB,max}-94}{20}} = 10^{\frac{-43+130-94}{20}} = 0.447V \quad (19)$$

As the microphone output is biased with a $0.8V$ offset, the maximum signal range of the microphone is about $0.353V - 1.247V$. To match this to the maximum ADC range of $0V - 3.3V$, the gain value for the low gain amplifier was chosen to be 2.5, to bring up the signal range to $0.883V - 3.117V$.

Using OKAWA Electric design tool to design a filter with $f_c = 100kHz, G = 2.5, \xi = 1$, resistors from the E6 series and capacitor from E24 series, the result design is as follow:

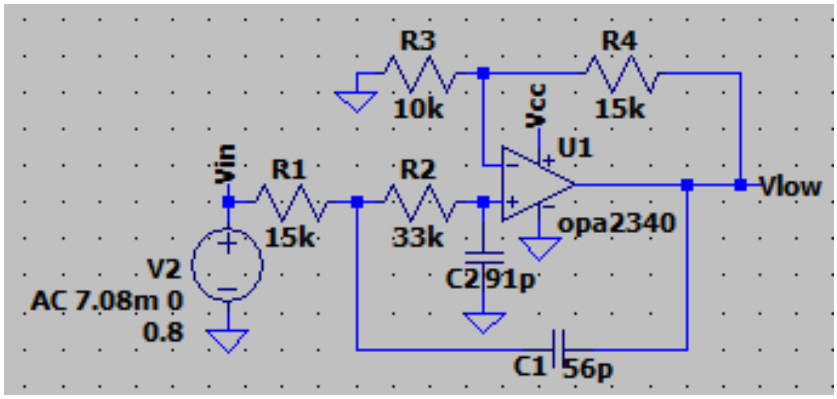

Figure 14: Sallen-Key filter and low gain amplifier design

The frequency response of this filter has a near $6dB$ attenuation at $100kHz$, as this is a second order filter. The $-3dB$ bandwidth of the filter is around $66.5kHz$.

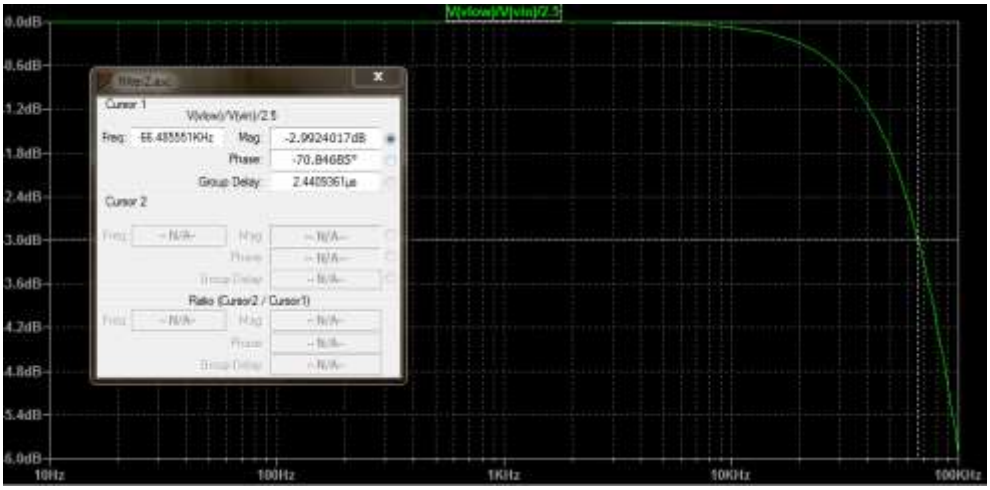

Figure 15: Frequency response of the anti-alias filter

The LSB error for the ADC can be calculated from the frequency response from /1/:

$$G = 20log\left(\frac{2^n - err}{2^n}\right)$$
$$\Rightarrow err = 2^n\left(1 - 10^{\frac{G}{20}}\right) \quad (20)$$

At $1kHz$, the attenuation of the filter is $-853\mu dB$, which corresponds to a LSB error of:

$$err = 2^{16}\left(1 - 10^{\frac{-853\times10^{-6}}{20}}\right) = 6.4\ LSB \quad (21)$$

Therefore, the ADC error for the amplifier is chosen to be 8 LSB. The lowest meaningful voltage value the ADC can measure from this amplifier is:

$$V = 8Q = 8 \times 50.35 \times 10^{-6} = 0.403mV \quad (22)$$

This corresponds to a sound level of (at $S_{dB,max} = -43dB$):

$$\begin{aligned} L_{dB} &= 20\log(V) - S_{dB} + 94 \\ &= 20\log(0.403 \times 10^{-3}) - (-43) + 94 \\ &= 69.1dB \end{aligned} \quad (23)$$

This sound pressure level $L_{dB} = 69.1dB$ is the lowest level can be measured from the low gain amplifier.

5.1.5 <u>High gain amplifier</u>

The high gain amplifier is a differential amplifier circuit. The main function of this amplifier is to bring up the signal in the low signal range to a level that the ADC can convert, and to move the DC offset of the signal to the middle of the $3.3V$ range of the ADC in order to best utilize the ADC dynamic range.

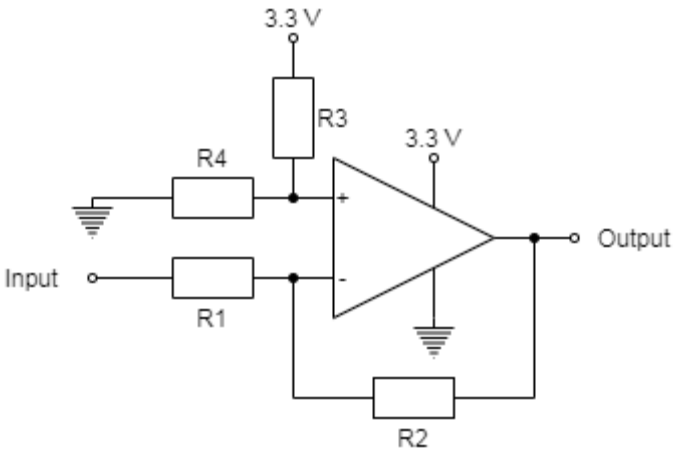

Figure 16: Differential amplifier circuit with opamp

The gain for the high gain amplifier should fulfill this inequation in order to bring the signal up to a sufficient level for the ADC:

$$V_{min} \times G_{low} \times G_{high} \geq err \times Q \tag{24}$$

With: V_{min}: minimum peak voltage signal output (V)

G_{low}, G_{high}: gain of the low and high gain amplifier, respectively

err: ADC error limit (8 LSB)

Q: ADC voltage resolution (V)

The minimum peak value signal the microphone may produce ($S_{dB,min} = -47dB$, $L_{db,min} = 30dB$) is:

$$V_{min} = 10^{\frac{S_{dB,min}+L_{dB,min}-94}{20}} = 10^{\frac{-47+30-94}{20}} = 2.82\mu V \tag{25}$$

We can calculate the gain for the high gain amplifier from (24):

$$G_{high} \geq \frac{err \times Q}{V_{min} \times G_{low}} = \frac{8 \times 50.35 \times 10^{-6}}{2.82 \times 10^{-6} \times 2.5} = 57.13 \tag{26}$$

Therefore, a gain of 57.31 was chosen for the high gain amplifier, as it can be implemented with two resistors from the E24 series, $47k\Omega$ and 82Ω.

The output value of the differential amplifier is /3/:

$$V_{out} = \frac{R_4}{R_3 + R_4} \times \frac{R_1 + R_2}{R_1} \times V_2 - \frac{R_2}{R_1} \times V_1 \tag{27}$$

Since $V_{ref} = V_2 \times \frac{R_4}{R_3+R_4}$, $G_{high} = \frac{R_2}{R_1}$, (27) becomes:

$$V_{out} = (G_{high} + 1) \times V_{ref} - G_{high} \times V_1 \tag{28}$$

$V_2 = 3.3V$ will be connected to the voltage supply. To get $V_{out} = \frac{V_c}{2} = \frac{3.3}{2} = 1.65V$ from $V_1 = 0.8 \times G_{low} = 0.8 \times 2.5 = 2V$:

$$V_{ref} = \frac{V_{out} + G_{high} \times V_1}{G_{high} + 1} = \frac{1.65 + 57.13 \times 2}{57.13 + 1} = 1.994V \tag{29}$$

Therefore, two resistors, $1.5k\Omega$ and $2.29k\Omega$ from the E192 series was chosen for the voltage divider.

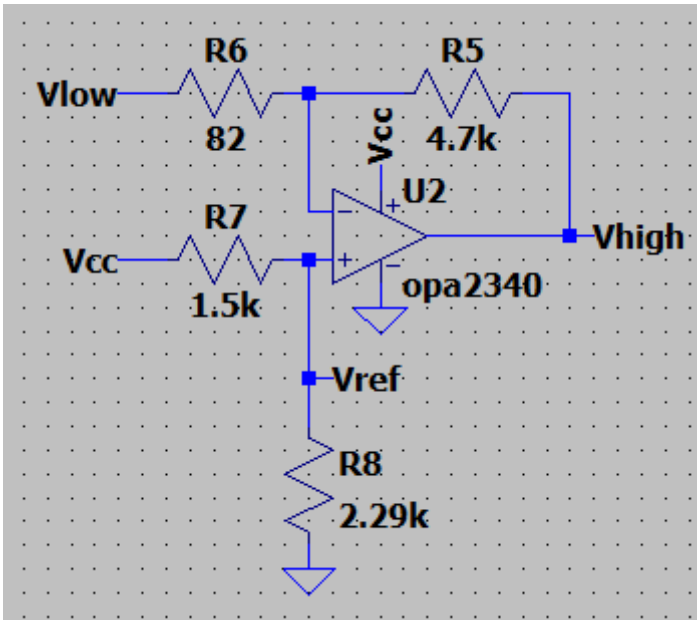

Figure 17: Differential amplifier circuit design

Note that due to the high gain of the amplifier, error from the resistor values from both of the amplifiers may push the offset of the high gain amplifier off the ADC range. As a result, resistors used for this design should have very low tolerate (at most 0.1%).

5.2 Software design

Due to a lack of equipment, the frequency response of the microphone could not be measured, and as a result, the equalizer part of the software was not implemented.

5.2.1 Toolchains

The firmware source code template for the device is generated by STM32CubeMX, and built with Atollic TRUEStudio for STM32.

5.2.2 ADC configuration

The device uses two channels CN15 and CN10 of the MCU's ADC1 for sampling in single ended mode. The two channels are respectively connected to pin A0 and A1 of the CN9 pin block on the Nucleo board. The ADC is configured to run in interrupt mode.

To sample the signal at 48*kHz*, the ADC is control by the MCU's timer TIM3. The timer is configured to use the internal clock and overflow at the sample rate. The trigger out event is selected to be Update Event in order to trigger the ADC.

Counter Settings	
Prescaler (PSC - 16 bits value)	8
Counter Mode	Up
Counter Period (AutoReload Register - 16 bits value)	101
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable
Trigger Output (TRGO) Parameters	
Master/Slave Mode (MSM bit)	Disable (Trigger input effect not delayed)
Trigger Event Selection TRGO	Update Event

Figure 18: TIM3 setting in the STM32CubeMX

The ADC1 is configured to run in discontinuous scan mode with two active channels. This means that the ADC will run two conversions on two channels one successively after the trigger event. The pin A0 (CN15) is connected to the low gain amplifier, while the pin A1 (CN10) is connected to the high gain amplifier. Channel CN10 is set to rank 1, and CN15 is set to rank 2. This means that the conversion for the high gain amplifier will be run first. The ADC will be trigger by the timer TIM3 trigger out event, rising edge. The conversion result is stored into the DR register, which will be read with the *HAL_ADC_GetValue* function. End of conversion is chosen to be triggered at the end of

each single conversion in order for the interrupt to be called after each channel conversion.

ADC_Settings	
Clock Prescaler	Asynchronous clock mode divided by 2
Resolution	ADC 16-bit resolution
Scan Conversion Mode	Enabled
Continuous Conversion Mode	Disabled
Discontinuous Conversion Mode	Enabled
Number Of Discontinuous Conversions	1
End Of Conversion Selection	End of single conversion
Overrun behaviour	Overrun data overwritten
Conversion Data Management Mode	Regular Conversion data stored in DR register only
Low Power Auto Wait	Disabled
ADC_Regular_ConversionMode	
Enable Regular Conversions	Enable
Left Bit Shift	No bit shift
Enable Regular Oversampling	Disable
Number Of Conversion	2
External Trigger Conversion Source	Timer 3 Trigger Out event
External Trigger Conversion Edge	Trigger detection on the rising edge
Rank	1
Channel	Channel 10
Sampling Time	1.5 Cycles
Offset Number	No offset
Rank	2
Channel	Channel 15
Sampling Time	1.5 Cycles
Offset Number	No offset

Figure 19: TIM3 setting in the STM32CubeMX

In the software, firstly, the ADC is calibrated with the *HAL_ADCEx_Calibration_Start* function to archive better results. Then the timer is started with *HAL_TIM_Base_Start*, and the ADC is started in interrupt mode with *HAL_ADC_Start_IT*.

```
// calibrate ADC
HAL_ADCEx_Calibration_Start(&hadc1, ADC_CALIB_OFFSET_LINEARITY,
ADC_SINGLE_ENDED);

// start timer
HAL_TIM_Base_Start(&htim3);

// clear GPIO terminal
HAL_UART_Transmit(&huart3, "\033[1J\033[H", 7, 100);

// start ADC
HAL_ADC_Start_IT(&hadc1);
```

Figure 20: ADC1 and TIM3 startup code (*main.c*)

The result will be collected when the ADC interrupt callback function *HAL_ADC_ConvCpltCallback* is called. A global variable *data_cn* is used to keep track of the current active channel.

```
void HAL_ADC_ConvCpltCallback(ADC_HandleTypeDef* hadc) {
    // read data
    uint32_t value = HAL_ADC_GetValue(hadc);

    // send data to normalizer
    if (data_cn == 0)
        NORM_HighData(&norm, value);
    else
        NORM_LowData(&norm, value);

    // update channel
    if (++data_cn >= 2)
        data_cn = 0;
}
```

Figure 21: ADC interrupt callback function (*main.c*)

The two function *NORM_HighData* and *NORM_LowData* will be discussed in more detail in the next section.

5.2.3 Channel select and normalizer

Channel selecting and normalizing is carried out by the structure *Normalizer* and its associating functions.

```
#define DATA float32_t

typedef struct Normalizer {
    void (*data_callback)(void* collector, DATA data);
    void* collector;
    DATA cache;
} Normalizer;

void NORM_Init(Normalizer* n);
void NORM_HighData(Normalizer* n, uint32_t data);
void NORM_LowData(Normalizer* n, uint32_t data);
```

Figure 22: *Normalizer* structure, associating functions and macro (*norm.h*)

The *Normalizer* structure has three fields:

- *cache*: used for channel select process
- *data_callback* function which will be called when a new piece of data is ready to be read

- *collector*: pointer to the collecter structure to be passed to the callback

It also has three associating functions:

- *NORM_Init:* initialize the structure by zero-ing all of its field
- *NORM_HighData*: read new reading for the high gain filter and save it to the cache
- *NORM_LowData*: read new reading for the low gain filter, pick the appropriate channel to avoid clipping data from the high gain amplifier

An instance of this structure is registered as a global variable in the main source file, and is initialized in the main function. Its association functions are call in the ADC interrupt callback. The collector for the normalizer is the buffer, which will be discussed in more detail in the next section.

5.2.4 Buffer

The buffer records the data and stores it for processing later to ensure that data capture happens in real time while the filter process can be done outside of the interrupt handle. This process is carried out by the structure *Buffer* and its associating function.

```
#define SAMPLE_COUNT 6000

typedef struct Buffer {
    void (*ready_callback)(DATA *data, size_t size);
    DATA buf[SAMPLE_COUNT * 2];
    size_t cur;
} Buffer;

void BUFF_Init(Buffer* b);
void BUFF_NewData(void* collector, DATA data);
```

Figure 23: *Buffer* structure, associating functions and macro (*buffer.h*)

The *Buffer* structure has three fields:

- *buf*: where the buffer data is stored, the size of this buffer is twice the size of the sample count, so one half of the buffer can be used for recording while the other half is processed in the main function.
- *cur*: the cursor pointing to the next position on the buffer to write data to
- *ready_callback*: call back function to be called when one half of the filter is ready for processing

It also has three associating functions:

- *BUFF_Init:* initialize the structure by zero-ing all of its field
- *BUFF_NewData*: callback function to be used by the normalizer

The SAMPLE_SIZE macro is set to collect 6,000 samples per RMS reading, or 8 readings per second for 48*kHz* sampling rate.

An instance of this structure is registered as a global variable in the main source file, and is initialized in the main function.

```
BUFF_Init(&buff);
buff.ready_callback = &callback;

NORM_Init(&norm);
norm.data_callback = &BUFF_NewData;
norm.collector = &buff;
```

Figure 24: Buffer and normalizer initialization in the main function (*main.c*)

The normalizer is set to use the buffer as its collector. The buffer's *ready_callback* is set to call a function in the main source file.

```
void callback(DATA *d, size_t size) {
    data = d;
    sz = size;
    flag = 1;
}
```

Figure 25: Buffer callback (*main.c*)

Note that all the callbacks so far are called in the interrupt handle context. Three global variables, *data*, *sz* and *flag* is used to send data from the interrupt handle to the main function in the buffer's callback.

5.2.5 A-weighting and RMS calculating

The A-weighting is carried out by the IIR filter implemented in the structure *Iir*.

```c
#define BLOCK_SIZE 400
#define STAGE_COUNT 3

typedef struct Iir {
    arm_biquad_casd_df1_inst_f32 inst;
    float32_t coeff[STAGE_COUNT * 5];
    float32_t state[STAGE_COUNT * 4];
    DATA result[BLOCK_SIZE];
} Iir;

void IIR_Init(Iir* f);
void IIR_SetCoeff(Iir* f, size_t pos, float32_t b1, float32_t b2, float32_t a1,
        float32_t a2);
void IIR_Process(Iir* f, DATA* data);
void IIR_RmsResult(Iir* f, float32_t* res);
```

Figure 26: *Iir* structure, associating functions and macro (*iir.h*)

The *Iir* structure has four fields:

- *inst, coeff, state*: structure and data buffer to be used in the IIR filter process, the size of *coeff* and *state* is dependent on the number of stages.
- *result*: data buffer where the result will be stored.
- *ready_callback*: call back function to be called when one half of the filter is ready for processing

It also has three associating functions:

- *IIR_Init:* initialize the IIR structure
- *IIR_SetCoeff*: set the coefficients for each IIR filter cascade stage, $b_0 = 1$
- *IIR_Process*: run the IIR filter process
- *IIR_RmsResult*: calculate the RMS of the current result buffer

The BLOCK_SIZE is the size of a block for the IIR filter, and this value must be a factor of the sample count.

The IIR filter is initialized and its coefficient is set according to (10).

```
IIR_Init(&filter);

float32_t b01 = 0.3;
float32_t b02 = 0;
float32_t a01 = -0.405;
float32_t a02 = 0.04100625;

float32_t b11 = -2.00013085;
float32_t b12 = 1.00013086;
float32_t a11 = -1.8957;
float32_t a12 = 0.8969642;

float32_t b21 = -1.99986915;
float32_t b22 = 0.99986916;
float32_t a21 = -1.9946;
float32_t a22 = 0.99460729;

IIR_SetCoeff(&filter, 0, b01, b02, a01, a02);
IIR_SetCoeff(&filter, 1, b11, b12, a11, a12);
IIR_SetCoeff(&filter, 2, b21, b22, a21, a22);
```

Figure 27: IIR filter initialization in main function (*main.c*)

The IIR filter is executed in the main while loop.

```
while (1) {
    HAL_Delay(1);

    if (flag > 0) {
        HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_SET);

        DATA blk[BLOCK_COUNT], rms;
        size_t i;

        // A-weighting and RMS calculate
        for (i = 0; i < BLOCK_COUNT; i++) {
            DATA* ptr = data + i * BLOCK_SIZE;
            IIR_Process(&filter, ptr);
            IIR_RmsResult(&filter, blk + i);
        }
        arm_rms_f32(blk, BLOCK_COUNT, &rms);

        // filter gain correct
        rms /= 1.64;

        // increase counter
        c += 1;

        // calculate dB value
        float db = 20 * log10(rms) + 139;

        sprintf(buffer, "#[%5.0f] %6.2fdB %10.4eV \r\n", c, db, rms);
        HAL_UART_Transmit(&huart3, buffer, strlen(buffer), 100);
        flag = 0;

        HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_RESET);
    }

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
}
```

Figure 28: Main while loop (*main.c*)

The main while loop will wait until the *flag* variable is set, indicating that new data is available in the buffer. If this is the case, the IIR filter will process the buffer data, block by block, and save the RMS results into the *blk* array. The *rms* value is then calculated by taking the RMS of all the blocks, and then modified with the gain correct. The gain correction value is calibrated for both the analog and digital filter to achieve a 0*dB* gain at 1*kHz* in order to comply with /4/. The dB value of the RMS then is calculated and corrected. The final value is then sent by UART to the PC's USB port via the on-board debugger.

6 EVALUATION AND RESULTS

Due to the COVID-19 pandemic, the testing for this design was limited by the available tools available at the student's resident. The hardware for this design is implemented on breadboard and connected to a Nucleo-H743ZI development board. A Digilent's Analog Discovery 2 USB oscilloscope was used as oscilloscope, signal generator and power supply.

Unfortunately, neither the Nucleo development board nor the Analog Discovery could provide a low noise DC voltage supply. This noise is equivalent to a 60*dB SPL* input from the microphone at the output of the high gain filter, rendering most analog analysis of the system invalid. Due to this, only general functionality of the design (detecting sound at a high enough volume, stable ambient noise level, linear correction, filter stability) was performed with satisfactory result. In order words, the design is able to give reasonable measure reading, but the precision of these readings cannot be determined.

Another factor of the design performance is the use of the CPU time for the filter process. This can be measured by setting and resetting an GPIO pin at the start and the end of this process.

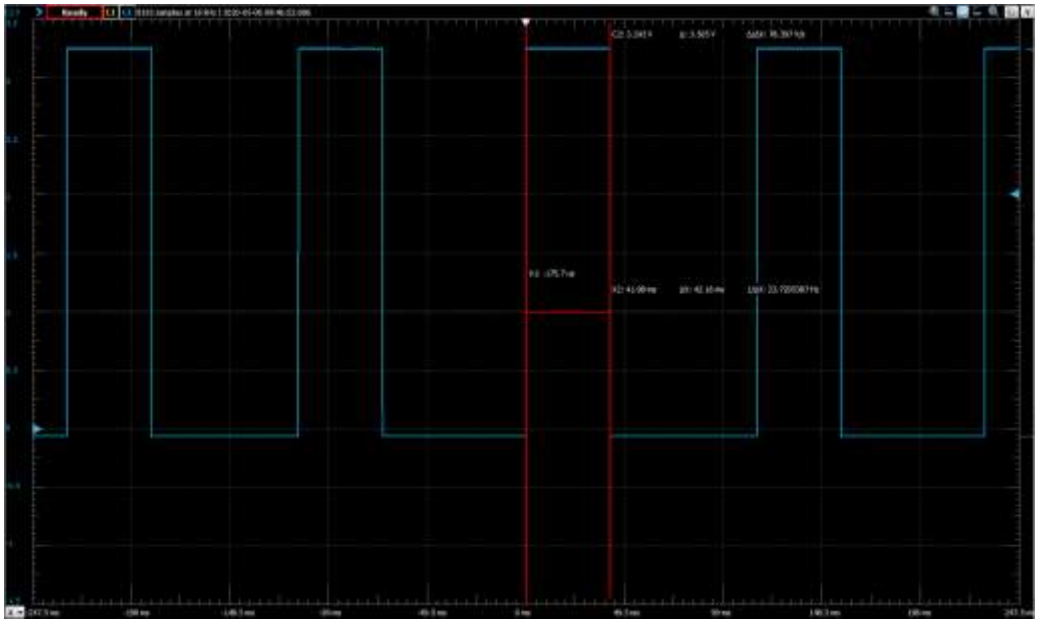

Figure 29: Filter process CPU time usage

The length of a cycle is the sampling period (125*ms*) while the length of the pulse is the CPU time used by the filter process (about 42*ms*, or 33.6% of the sampling period). This shows that there may be enough resources for the equalizer and wireless communication

to be added later. Note that the implementation of the CMSIS DSP library used for this test did not fully utilize the capability of the MCU DSP instructions, so with proper optimization, the result may be improved.

7 CONCLUSION

After the project, the requirement for the IEC-67172:2013 was thoroughly inspected and a proof-of-concept design was created, showing that a design for a sound level measuring device can be satisfactory with the filter process done mainly by the software. However, more work needs to be done to test and calibrate the design. There are some problems with the design that was not yet address, such as DC voltage supply/reference, CMSIS DSP usage, ADC linear discrepancy.

8 GLOSSARY

IEC	International Electrotechnical Commission
MEMS	Microelectromechanical System
CMSIS	Cortex Microcontroller Software Interface Standard
DSP	Digital Signal Processing
ASIC	Application-Specific Integrated Circuit
SPL	Sound Presure Level
OpAmp	Operational Amplifier
ADC	Analog-to-Digital Converter
FPU	Floating-point Unit
RAM	Random Access Memory
MSPS	Mega-samples Per Second
FIR	Finite Impulse Response
IIR	Infinite Impulse Response
API	Application Programming Interface
MCU	Microcontroller Unit
RTOS	Real-time Operating System
SIMD	Single Instruction, Multiple Data
LSB	Least-significant bit
DC	Direct Current
RMS	Root Mean Square
UART	Universal Asynchronous Receiver/Transmitter

9 REFERENCES

/1/ Baker, B. C. 2015. Designing an anti-aliasing filter for ADCs in the frequency domain. Analog Applications Journal, 7-9. Accessed 03.03.2020. http://www.ti.com/lit/an/slyt626/slyt626.pdf?ts=1590970290152

/2/ Hauden, D. 2017. Microsystems to Nano-Microsystems: A Technological Breakthrough. In MEMS: Fundamental Technology and Applications 3-18. V. Choudhary, & K. Iniewski. (Eds.) Boca Raton, FL. CRC Press.

/3/ Holt, H. 2014. A Deeper Look into Difference Amplifiers. Analog Dialog 48, 1. Accessed 03.03.2020. https://www.analog.com/media/en/analog-dialogue/volume-48/number-1/articles/deeper-look-into-difference-amplifiers.pdf

/4/ IEC 61672-1. Electroacoustics - Sound level meters - Part 1: Specifications. IEC 61672-1:2013. IEC. 2013. 50

/5/ Lewis, J. 2013. Analog and Digital MEMS Microphone Design Considerations. Accessed 04.02.2020. https://www.analog.com/media/en/technical-documentation/technical-articles/Analog-and-Digital-MEMS-Microphone-Design-Considerations-MS-2472.pdf

/6/ Malcovati, P., Grassi, M., & Baschirotto, A. 2013. Interface Circuits for MEMS Microphones. In Nyquist AD Converters, Sensor Interfaces, and Robustness. A. H. van Roermund, A. Baschirotto, & M. Steyaert. (Eds.). New York, NY. Springer.

/7/ OKAWA Electric Design. 2020. Sallen-Key Low-pass Filter Design Tool. Accessed 21.02.2020. http://sim.okawa-denshi.jp/en/OPseikiLowkeisan.htm

/8/ Rimell, A. N., Mansfield, N. J., & Paddan, G. S. 2015. Design of digital filters for frequency weightings (A and C) required for risk assessments of workers exposed to noise. Industrial Health, 53(1), 21-27. Accessed 14.02.2020. doi:10.2486/indhealth.2013-0003

/9/ Risojević, V., Rozman, R., Pilipović, R., Češnovar, R., & Bulić, P. 2018. Accurate Indoor Sound Level Measurement on a Low-Power and Low-Cost Wireless Sensor Node. Sensors (Basel, Switzerland), 18(7). Accessed 12.02.2020. doi:10.3390/s18072351

/10/ Robinson, D. P., & Tingay, J. 2014. Comparative study of the performance of smartphone-based sound level meter apps, with and without the application of a 1/2″ IEC-61094-4 working standard microphone, to IEC-61672 standard metering equip-

ment in the detection of various problematic workplace. Inter Noise Conference. Melbourne, Australia. Accessed 26.02.2020. https://www.acoustics.asn.au/conference_proceedings/INTERNOISE2014/papers/p565.pdf

/11/ Shah, M. A., Shah, I. A., Lee, D.-G., & Hur, S. 2019. Design Approaches of MEMS Microphones for Enhanced Performance. Journal of Sensors, 1-26. Accessed 22.02.2020. https://doi.org/10.1155/2019/9294528.

/12/ Hakala, I., Kivelä, I., Ihalainen, J., Luomala, J., Gao, C. 2010. Design of Low-Cost Noise Measurement Sensor Network: Sensor Function Design. Accessed 22.02.2020. https://www.researchgate.net/publication/228655741_Design_of_Low-Cost_Noise_Measurement_Sensor_Network_Sensor_Function_Design

/13/ ICS-40300 Datasheet. Accessed 17.02.2020. https://invensense.tdk.com/wp-content/uploads/2019/02/DS-ICS-40300-00-v1.3.pdf

/14/ CMSIS DSP Software Library Reference. Accessed 15.04.2020. http://www.keil.com/pack/doc/CMSIS/DSP/html/index.html

/15/ OPAx340 Datasheet. Accessed 26.04.2020. http://www.ti.com/lit/ds/symlink/opa2340.pdf?ts=1590971142568

/16/ DS12110. 32-bit Arm® Cortex®-M7 480MHz MCUs, up to 2MB Flash, up to 1MB RAM, 46 com. and analog interfaces. Accessed 14.04.2020. https://www.st.com/resource/en/datasheet/stm32h743vi.pdf.

/17/ RM0433. STM32H742, STM32H743/753 and STM32H750 Value line advanced Arm®-based 32-bit MCUs. Accessed 14.04.2020. https://www.st.com/resource/en/reference_manual/dm00314099-stm32h742-stm32h743753-and-stm32h750-value-line-advanced-armbased-32bit-mcus-stmicroelectronics.pdf

/18/ AN4426. Tutorial for MEMS microphones. Accessed 04.02.2020. https://www.st.com/content/ccc/resource/technical/document/application_note/46/0b/3e/74/cf/fb/4b/13/DM00103199.pdf/files/DM00103199.pdf/jcr:content/translations/en.DM00103199.pdf