Real-time Sound Analysis to Count Opening Cycles of Automatic Doors

TEODOR RENMAN
CHARLIE RINGSTRÖM
MASTER´S THESIS
DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Real-time Sound Analysis to Count Opening Cycles of Automatic Doors

Teodor Renman
Charlie Ringström

Department of Electrical and Information Technology
Lund University

Supervisor: Maria Sandsten

Examiner: Thomas Johansson

June 20, 2023

Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

Counting opening cycles on an automatic sliding door is of great interest for a company manufacturing doors, such as ASSA Abloy. These metrics could be used for consumer statistics or for door diagnostics. Counting opening cycles is seemingly trivial when there is access to the door's internal diagnostics or having adequate sensors. Problems start to arise when these are not present, which is the case when working with third party door vendors, and sensors can often be expensive and time consuming to install.

This report investigates the process of counting opening cycles using real-time sound analysis on a microcontroller. Due to microphones having low cost and lack need for precise placement, ASSA Abloy is considering implementing this on their IoT gateway hardware, ready to be mounted on any third party door.

To achieve this, two different algorithms were tested - signal energy analysis and Mel Frequency Cepstrum Coefficient (MFCC) analysis. An implementation was also tested were both algorithms were used. These were then tested on four different kinds of automatic doors.

Signal energy detection performs well on all types of doors but is prone to false positives from external noise. MFCC detection is more resistant to false positives from a noisy environment, but often detects false positives during the closing of the door, and only gives good results for two doors. The combined algorithm similarly only performed well for two of the doors, but had the fewest false positives. These results show that in an ideal environment signal energy detection will suffice, but in noisy scenarios the combined algorithms show promise, assuming the MFCC detection can be improved or an initial calibration to the door is added.

Popular Science Summary

Using sound analysis to count automatic door opening cycles

Counting door cycles is useful for a company manufacturing automatic doors, especially if there is a solution that works for any automatic door. Different methods of sound analysis were tested on different types of doors to determine how best to detect opening cycles for any door.

Counting how many times an automatic door opens - the number of opening cycles, is a seemingly trivial task. For a door manufacturer, for example ASSA Abloy, this task is simple, as this information can be accessed with the internal electronics of the door. In certain cases, such as if ASSA Abloy needed to do maintenance on a door from a different company, this might not be possible. This can be problematic if usage statistics are desired by a customer, or if ASSA Abloy is contracted to perform maintenance for an entire door building containing various types of doors and want to use opening cycles as a diagnostic tool. Because of this, the need for external sensors arises. Placing sensors to detect when the door is opening can however be time consuming and expensive when doing it for many doors.

This is why this paper investigates the use of microphones for listening for door openings instead. Using microphones has two big advantages - low cost and ease of installation. Although an algorithm for doing this is inherently more complex to develop, it has the potential to reduce cost when applied at scale. The sound analysis is done on a microphone connected to a microcontroller which analyses the audio in real-time. By recording the sound of multiple opening cycles from different doors suitable parameters could be programmed into the microcontroller when tuning the algorithms.

Two different methods were tested - signal energy analysis and Mel Frequency Cepstrum Coefficient (MFCC) analysis. The signal energy analysis is a very simple algorithm detecting sustained periods of high signal energy - prone to many false positives from background noise, but rarely misses an opening cycle. The other algorithm is more sophisticated and is capable of differentiating between several different kinds of sound signatures. The MFCC algorithm is designed to mimic how humans interpret sound. This means that the difference in the MFCC output varies distinctly when for example a human says different vowels, which is why this algorithm is often used in speech processing, but has proven to be very useful when detecting door opening sounds as well. A

downside to this algorithm is that they detect both the opening and closing of the doors, when only the door opening should be detected.

The energy-based algorithm worked best in quiet environments, while the MFCC-based algorithm only performed well on specific doors. This is due to how different the sound is between different doors, and it is difficult to make a generalised algorithm which works between multiple door types while not generating false positives. Using both algorithms at the same time also only had good results for half of the doors, but was the most resistant to false positives from background noise. In more realistic environments, the combined algorithms may be the most useful, if the MFCC algorithm can be improved for all doors, or if the microcontroller was calibrated for a doors opening sound before being put to use.

Foreword

We would like to thank ASSA Abloy for proposing the project idea, as well as supplying hardware and doors to test on. Among the many helpful staff members of ASSA Abloy, we in particular thank Roger Dreyer, Mathias Navne, Fredrik Brodje, Louise Bannersten, Elin Elfström for their assistance and support over the course of the project.

We give thanks to our supervisor Maria Sandsten, who through extraneous circumstances went above and beyond our expectations, constantly helping us make progress with the project and the report. We also thank Christoffer Cederberg for designing the circuit used for the project.

Finally, we would like to thank our families and friends, for their support and encouragement over the last 6 months of work.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

To count the number of opening cycles of a door is a seemingly trivial task which has many use cases, such as detecting when a store has the most amount of visitors or simply for testing how many times the door can open and close before it breaks. Doing this is very simple when having access to the door's internal diagnostics, but sometimes the need for counting opening cycles for third party doors arises, as is the case for ASSA Abloy.

There are many ways of solving this problem, many of which share a common problem - installation cost. This paper proposes an alternative solution to this problem - audio detection. Although this solution is more complex to develop, it has the potential to reduce installation time and cost.

ASSA Abloy has expressed interest in including a microphone in their IoT Gateway, which is an internet connected device used internally in many of ASSA Abloy's systems and is usually mounted in the in the upper corner of their doors, which is ideal for sound analysis.

1.1 Project aims and challenges

The goal of this Master's thesis work is to create a solution to perform sound analysis that can be used to count the number of opening cycles an automatic door performs, which can then be used on ASSA Abloy's IoT gateway platform. ASSA Abloy is interested in using the solution on many different models of automatic doors, including doors not produced by them. The function is not critical to the operation of the doors, meaning a missed opening cycle or a false positive will not have serious consequences. Despite this, ASSA Abloy naturally wants the cycle detection to have a high degree of accuracy. These two facts mean a key part of the project is determining which algorithm for sound analysis can be flexible enough to detect opening cycles from various types of doors, as well as being reliable enough to not give false positives when conditions such as external noise are present.

Due to hardware restrictions, the code for real-time analysis cannot be written with Python or Matlab as is commonly done when performing sound analysis, and has to be done with C code instead. C can be less forgiving and more strict when it comes to memory management and data types than Python/Matlab, and more code has to be written in order to perform the same task, but has the advantage of being lower level and

being able to achieve higher performance on weak systems, such as a microcontroller. Since ASSA Abloy is considering implementing this algorithm on their IoT gateway, performance needs to be taken into account, since it needs to be able to run other tasks simultaneously.

1.1.1 Defining an opening cycle

According to ASSA Abloy, an opening cycle is defined as "when an automatic door transitions from a closed or semi-closed state to a fully open state". This means that the algorithm should only detect an opening cycle when the door is opening, and not when it is closing. It is a challenge to differentiate opening sounds from closing sounds due to how similar they are in nature. As an initial constraint for the project, all automatic doors are assumed to be fully closed before opening.

1.1.2 Machine learning and event detection

In the field of audio event detection, the approach of using machine learning on a data set of audio clips to generate detection models has become highly prevalent.[10] While this can often produce highly accurate results for detecting specific sounds,[21] it has not been used for this project. Most research using machine learning for audio event detection do not have hardware constraints as strict as is the case here, and they usually do not perform the analysis in real time. This makes previous results less relevant for this project. Although some efforts have been made to use machine learning in real time with restricted processing power,[16] the method is ultimately most useful when having to identify and differentiate between many different sounds. Since this project only requires identifying a specific event there are other suitable approaches that are simpler to implement, which is appropriate given the restrictions of the project.

Chapter 2

Theory

In order to understand the findings of the project, cursory knowledge of the hardware and software used, as well as signal processing, is required. The necessary concepts are explained in this chapter.

2.1 Converting audio to digital data

Audio is sound expressed as an analogue signal. In order to store and process audio the signal has to be converted to a digital signal. Since an analogue signal is continuous it has to be sampled at discrete points. The Nyquist-Shannon theorem states that in order to discretise a signal without loss of information, the sampling rate has to be twice that of the highest frequency contained in the original signal. For this project, a sampling frequency of 32 000 Hz is used.

2.1.1 DMA

Direct Memory Access (DMA) is a feature found in most microcontrollers and is used for sending and receiving digital data. Since receiving data is relatively time consuming, this task is ideally not done directly on the CPU and is instead handed over to the DMA, which writes to a certain section in memory independently from the CPU, allowing it to perform other tasks while waiting. When DMA has finished receiving data it calls an interrupt to let the program know that a chunk of data is ready to be processed. The same also goes for sending data and is used for writing sound data to an SD card using the 1 bit SDIO protocol.

When the DMA interrupts the microcontroller to indicate that data has been written to memory it immediately starts overwriting new data. This can cause the microcontroller to process data which has been partially overwritten, leading to undesired results. Because of this, a double buffer system is introduced. This means that another interrupt function is called when half of the data has been written to memory, giving the program enough time to process that section.[6]

2.2 Signal Theory

2.2.1 Infinite Impulse Response (IIR) filter

For filtering the sound data an IIR filter is used, which has the equation:[17]

$$0 = \sum_{k=0}^{N} b[k]x[n-k] - \sum_{k=0}^{P} a[k]y[n-k] \tag{2.1}$$

where

- $x[k]$ is the input signal
- $y[k]$ is the filtered output signal
- a and b are filter coefficients

The IIR filter can be configured to be either a highpass or a lowpass filter depending of the values of a and b. These values are typically generated with tools such as `scipy.signal.butter`[19] in Python or Matlab's `butter` function.[9]

2.2.2 Digital biquad filter

The IIR filter implementation used on the microcontroller is called the biquad cascade filter, which is a second order IIR filter implementation. Biquad cascade filters comes in several different variants, with *direct form 1* (DF1) and transposed direct form 2 (TDF2) being of relevance in this paper.

DF1 has the equation

$$y[n] = \frac{1}{a_0}\big(b_0x[n] + b_1x[n-1] + b_2x[n-2] - a_1y[n-1] - a_2y[n-2]\big) \tag{2.2}$$

with the transfer function

$$H(z) = \frac{b_0 + b_1z^{-1} + b_2z^{-2}}{a_0 + a_1z^{-1} + a_2z^{-2}} \tag{2.3}$$

and can be represented with the signal flow graph in figure 2.1.

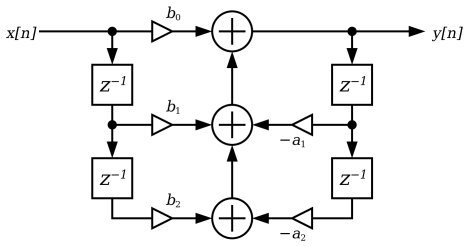


Figure 2.1: Signal flow graph of a biquad cascade filter in direct form 1[2]

Equation (2.2) can be rewritten as

$$\begin{cases} y[n] = b_0x[n] + s_1[n-1] \\ s_1[n] = s_2[n-1] + b_1x[n] - a_1y[n] \\ s_2[n] = b_2x[n] - a_2y[n] \end{cases} \tag{2.4}$$

and is referred to as its *transposed direct form 2* (TDF2) and keeps the same transfer function in (2.3) and is represented with the signal flow chart in figure 2.2

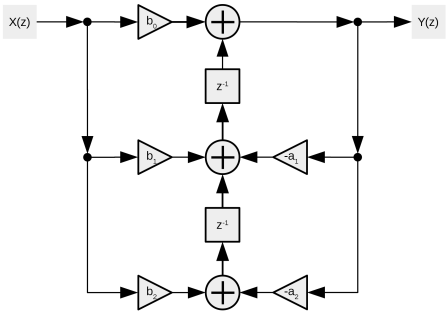


Figure 2.2: Signal flow graph of a biquad cascade filter in transposed direct form 2[14]

For input signals represented in floating point notation the TDF2 is preferred, since it is less susceptible to small float errors being accumulated. For a signal represented in fixed point notation (mentioned in more detail in appendix A) DF1 is most efficient.[15]

2.2.3 Fourier Transform

A Fourier transform is a transform that converts a signal from being described as a function of time to a function of frequency. This can be used to determine at what intensities various frequencies are prevalent in a signal based on the magnitude of the transform, as well as the phase based on the imaginary term. Fourier transforms can be performed with

$$X(f) = \int_{-\infty}^{\infty} x(t) \cdot e^{-i2\pi ft}\, dt \tag{2.5}$$

where

- $x(t)$ is the input signal
- $X(f)$ is the Fourier transform.

Because the signal is recorded as discrete data by the microcontroller, a discrete Fourier transform (DFT) algorithm must be used in place of the continuous Fourier transform, which can be performed with [5]

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-\frac{i2\pi}{N} kn} \tag{2.6}$$

where

- $X[k]$ is the frequency component of index k
- $x[n]$ is the input signal.
- N is the total number of samples, as well as the number of output frequency bins (unless the input signal is zero padded)

There exists a number of different algorithms for performing DFTs which are computationally faster, but give the same result. These go under the umbrella term Fast Fourier Transforms, or FFTs, which are typically used in place of DFTs.

If the input data is purely real, which is the case when recording audio, the resulting transformed output $X[k]$ is conjugate symmetric:

$$X[N-k] = X^*[k] \tag{2.7}$$

where X^* is the complex conjugate of X. This symmetry can be utilized in an FFT algorithm in order to save roughly half the calculations necessary to compute the transform.[4]

2.2.4 Mel Frequency Cepstral Coefficients (MFCC)

One way of analysing audio is with MFCC. MFCC stands for Mel Frequency Cepstral Coefficients. The MFCC contain information about the audio frequencies in a way more similar to how humans perceive them. To calculate the MFCC of a signal a discrete cosine transform (DCT) is performed on the logarithm of the short term energy spectrum expressed in the Mel-frequency scale.[1]

Before calculating the MFCC for the signal, the signal first needs to be split into short frames. Then a window function is applied to each frame to avoid spectral leakage. To calculate MFCC the following steps are performed:

1. Perform DFT for each frame.
2. Calculate the periodogram from the DFT.
3. Apply Mel filter banks to the periodogram.
4. Calculate the logarithm of each filter bank.
5. Calculate the discrete cosine transform (DCT) from the filter banks.

After the MFCC is calculated, a lifter (e.g. filter for MFCC) is implemented to the coefficients to suppress noise. These steps will be explained in more detail in the following sections.

Framing and windowing

Due to how the hardware records and processes the data in real time there is no need to perform any framing of the data. Windowing is performed using a Hamming window, which is defined as

$$w(n) = 0.54 - 0.46 \cdot \cos\left(\frac{2\pi n}{N-1}\right) \quad 0 \leq n < N \tag{2.8}$$

for N number of samples.[20]

Dividing a signal into frames could for example split a periodic signal in the beginning of a cycle and end on the end of a cycle. This gives rise to errors called ”spectral leakage” when performing DFT, which is why window functions are used to ”fade in” and ”fade out” the signal.

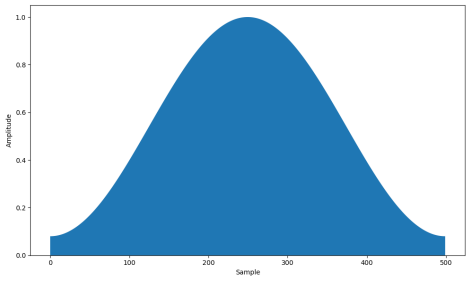


Figure 2.3: Hamming window

Periodogram

After performing a DFT of the signal frame, the periodogram is calculated using

$$P[k] = \frac{1}{N}|X[k]| \qquad 0 \leq k < N \tag{2.9}$$

where N is the number of frequency bins from the FFT. (as well as the number of samples per frame unless it is zero padded)

Mel frequency filter banks

The Mel frequency scale is a subjective scale of pitches designed to mimic how humans perceive pitch. To convert from frequency to the Mel scale, this formula is commonly used:[11]

$$m = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right) \tag{2.10}$$

Since the scale is subjective, the equation used may vary slightly.

The Mel filter banks consists of overlapping triangular sections shown in figure 2.4. The triangle end points are equidistant to each other in the Mel scale, but not in the frequency scale. Outside of these end points the filters consists of zeroes. These values are then converted back to the frequency scale with

$$f = 700 \cdot (10^{(m/2595)} - 1) \tag{2.11}$$

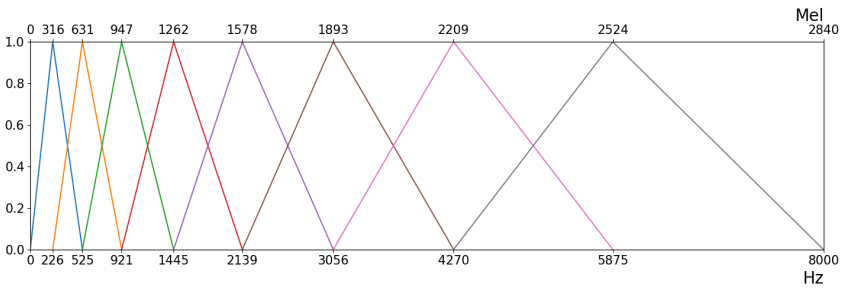


Figure 2.4: Mel filter bank with 8 filters.

Each value in the periodogram is multiplied with the filter for their respective frequencies and the result is then added up. A filter bank with K number of filters would generate K different values, each containing information about the power for their respective frequency bin in the Mel scale. The logarithm is then calculated for each of these values.

Discrete Cosine Transform (DCT)

The discrete cosine transform is similar to the DFT in that it breaks down the signal to its frequency components.

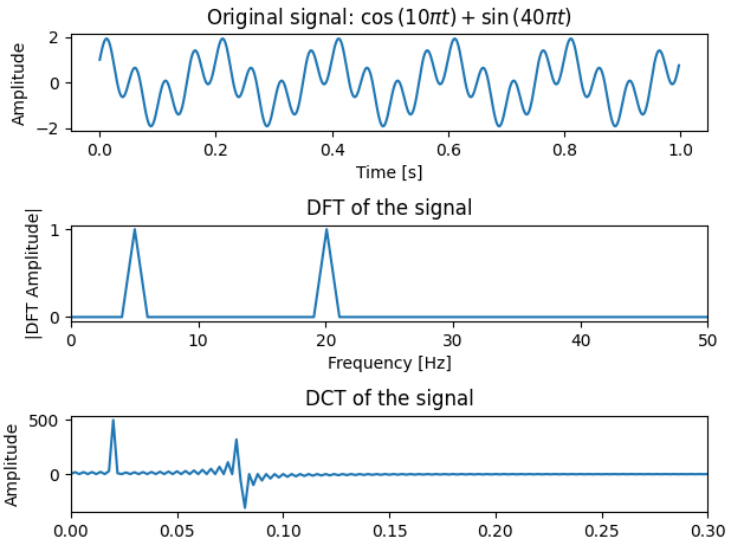


Figure 2.5: DFT and DCT comparison

Similar to the DFT, the original signal can be reconstructed from the DCT. Unlike the DFT, the DCT can only be applied to real-valued signals.

There exists multiple types of DCT equations, each with slight difference in their output, with the most common being the DCT-II variant: [18]

$$D[n] = \sum_{k=0}^{K-1} X[k] \cdot \cos\left(\frac{\pi}{K}\left(k + \frac{1}{2}\right)n\right) \quad 0 \leq k < N \tag{2.12}$$

where:

- $D[n]$ is the MFCC at index k
- $X[k]$ is the value at the frequency bin of index k from the FFT
- K is the number of frequency bins from the FFT

This equation is quite similar to the DFT equation in (2.6), and the result from this step can be interpreted as "the frequency of the frequency of the original signal".

DCT is performed on the result from the Mel filter banks. If the filter banks contains k = 25 filters the DCT will also contain 25 output values. Since higher values in the DCT represents higher frequencies, these are frequently left unused. For speech processing, there are usually 25 Mel filters with the first 13 of them being used.[7]

Finally, a sinusoidal lifter is applied to the DCT output, which has the equation

$$M[n] = D[n]\left(1 + \frac{L}{2} * \sin\left(\pi * \frac{n}{L}\right)\right) \tag{2.13}$$

where

- $M[n]$ is the liftered signal
- L is the liftering constant

This is used to suppress fast slow variations in the log power spectrum.[13]. For this project $L = 22$ is used.

Chapter 3

Method

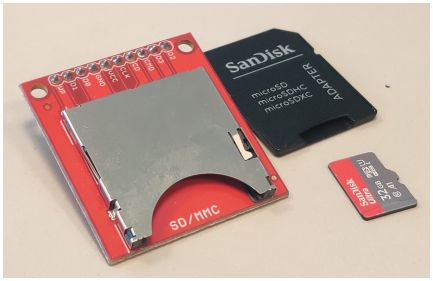
The following chapter explains what hardware and software was used in order to record opening cycles to study, as well as for performing analysis in real time.

3.1 Hardware

ASSA Abloy originally intended the thesis to make use of their own in-house developed circuit called the IoT Gateway used internally for several different products. Due to some difficulties in getting the I2S communication protocol used by the microphone to work properly with their build system it was decided to make use of another microcontroller board instead. The STM32 Blackpill board made by Nanjing Micro One Electronics Inc was chosen due to its STM32 F411 microcontroller being similar to the F407 on the IoT Gateway, being fairly inexpensive and because of its popularity among hobbyists.

The SPH0645 digital MEMS microphone made by Knowles Electronics was used for the project due to being fairly inexpensive, being capable of sending digital sound data and for having an easy to use breakout board made by Adafruit. To log data to the SD card a simple SD card breakout board was used. Both of these components are shown in figure 3.1.

(a) SPH0645 microphone breakout board.



(b) SD card breakout board alongside the actual memory card.

Figure 3.1: Components connected to the microcontroller for the project.

3.1.1 Hardware integration

The SPH0645 microphone is connected to the microcontroller via three pins - the clock pin, the word select pin and the serial data pin. Since I2S was designed to be a stereo protocol, the word select pin decides whether the left or the right microphone should output data. Through a pull-down resistor the SPH065 microphone is configured to only send data on the left channel, meaning that half the time no data is sent. This data is then stored in memory using DMA.

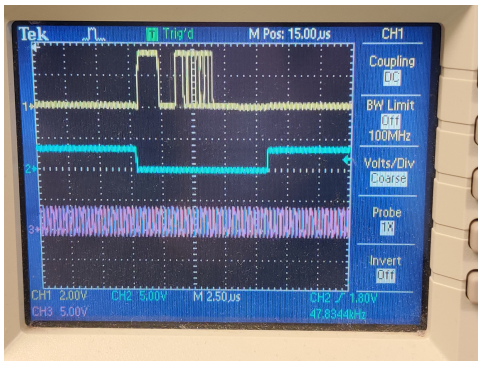


Figure 3.2: The SPH0645 microphone output on the oscilloscope.

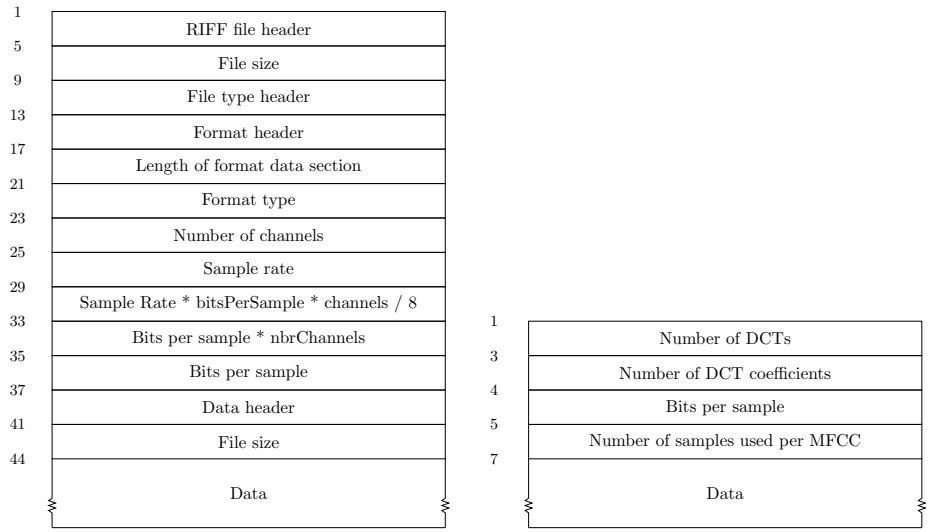
The SPH0645 sends 24 bit data on a 32 bit frame, meaning that the 8 last bits are set as 0. Despite this, when first testing the microphone the last 8 bits of the second integer would often contain stray bits. This was solved by raising the output speed of the clock pin.

For logging the data to the SD card the 1-bit SDIO protocol is used, with information sent using DMA as well. To prevent the microphone recording new data faster than packets of data were being saved to the SD card, fewer large packets with data were sent to the SD card instead of multiple smaller packets in order to increase performance.

The data is sent with the .WAV file format. This file consists of a header with information such as bit rate and the length of the file, and the rest consists of 32 bit sound data. When logging sound, the data is stored in the WAV-file continuously, and the header is edited once logging is complete to add the correct length of the recording. For debugging purposes, other data were also stored on the SD card containing MFCC output data with a custom made header. The contents of these headers are displayed in figure 3.3.

The algorithm was first written in Python in order to test and visualise the results, which later became the basis for the more efficient C implementation for the microcontroller. To program and debug the microcontroller an ST-Link/V2 was used. All of these components formed two different configurations used in the project. The first is a detector, containing a microcontroller, a microphone, and a programmer. This construction aims to replicate the hardware that will be used in the final solution for detecting opening cycles. This setup can be seen in figure 3.4.

The second setup is a logger, featuring the same components as the detector, but with the additional SD card reader. This setup was used in order to record and analyze opening cycles with the microphone used in the detector. This configuration is shown in figure 3.5



(a) WAV header format. **(b)** Custom header format for logging MFCC data.

Figure 3.3: Two headers used for logging.

3.2 Analysis of automatic doors

The development of the algorithm can be divided into two sections. The first is a pre-study, in which opening cycles were recorded using a microcontroller and examined on stronger hardware in order to create suitable detection algorithms. The latter section was the resulting algorithms being implemented on the microcontroller to perform the analysis in real time. As the solution was meant to be used on various types of automatic doors, both sections made use of four different doors named A-D, and are described in figure 3.6. For the pre-study, the logger was used to record opening cycles from each of the doors, while the detector was used on the doors for testing the results in real time. In all cases, the surroundings of the doors were mostly quiet, and the doors always started from a fully closed position before opening and closing.

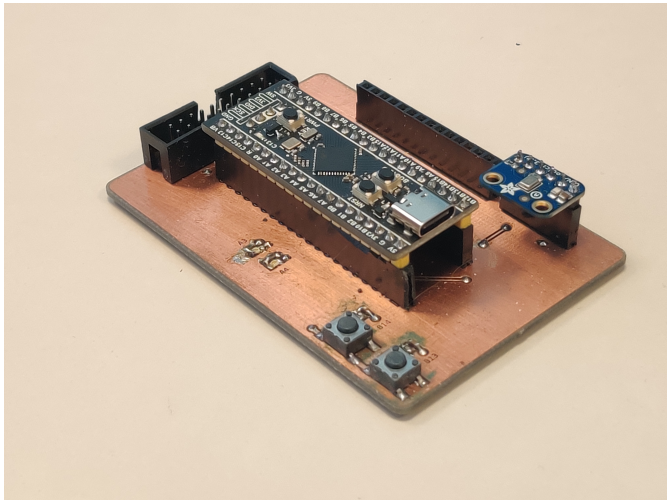


Figure 3.4: The setup for developing the detector, consisting of a microcontroller, a microphone and a PCB connecting them.

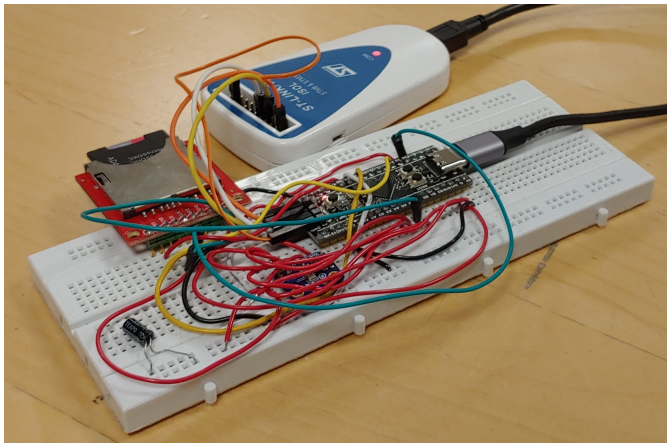
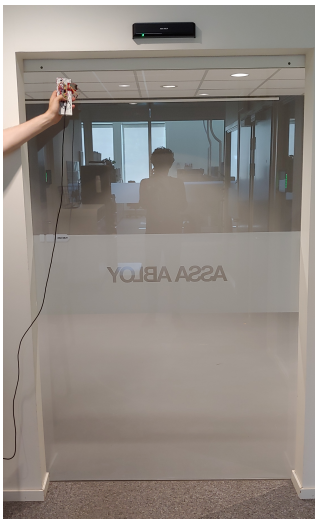


Figure 3.5: The setup for recording and logging audio data, using a microcontroller, a microphone, an SD card reader and a programmer.

(a) Door A. A locked automatic door that unlocks and opens using a key card. When the door unlocks an audible "click" sound can be heard.

(b) Door B. An automatic double sliding door that opens two doors when the sensor detects someone in front of it.

(c) Door C. An automatic double sliding door that unlocks and opens using a key card. A "click" can be heard when unlocking, but not as loud as door A.



(d) Door D. An automatic door that opens when the sensor detects someone in front of it. In all recordings the logger was approximately positioned as can be seen in the top left.

Figure 3.6: The different doors used for testing.

Chapter 4

Pre-study

In order to determine which algorithms were suitable for detecting opening cycles, audio from several different types of doors were recorded using the logger configuration and analysed in Python. 15 door cycles for each of the 4 door types were recorded, creating 60 audio clips that were transferred to a computer for analysis. This chapter details how the data was processed and analysed, what characteristics of the opening cycles were discovered, and what algorithms could be used to detect these characteristics.

4.1 Post-recording processing

Due to a low frequency offset present in in all recordings produced by the microphone, a highpass filter was applied with cutoff frequency $f_c = 100\,\text{Hz}$. At this point the audio could be analysed, but in order to make the WAV-file listenable the signals had to be scaled to fit a 16-bit integer. The listenable WAV-files were used to verify the validity of the recordings, but the unscaled versions were used for analysis.

In order to simplify analysis, each recording was also zero-padded at the start and end of each clip so that all opening cycles started at the same time and that all clips were of the same length. This means every opening cycle is expected to occur between seconds 1-4 in every clip. Any detection occurring outside of this interval should be regarded as a false positive.

4.2 Results of pre-study

The processed audio clips were read as data in Python, where various audio processing transforms and methods were used in order to determine common features between all opening cycles.

4.2.1 Periodogram

Periodograms, showing the average spectral density of the opening cycles, were created by performing the Fourier transform on each recording over the interval in which the door opening occurs. The results can be seen in figure 4.1, along with a reference recording showing the section in which the opening sound occurs. Because there were no noticeable higher frequencies present in the periodogram, only the results from 0 - 4 KHz are shown. The figure shows that there are mostly frequencies below 1000 Hz present during the

opening of the door. Due to the fact that there were no clear frequency spikes that could be differentiated from most other audio clips, the periodograms were not used further when developing detection algorithms.

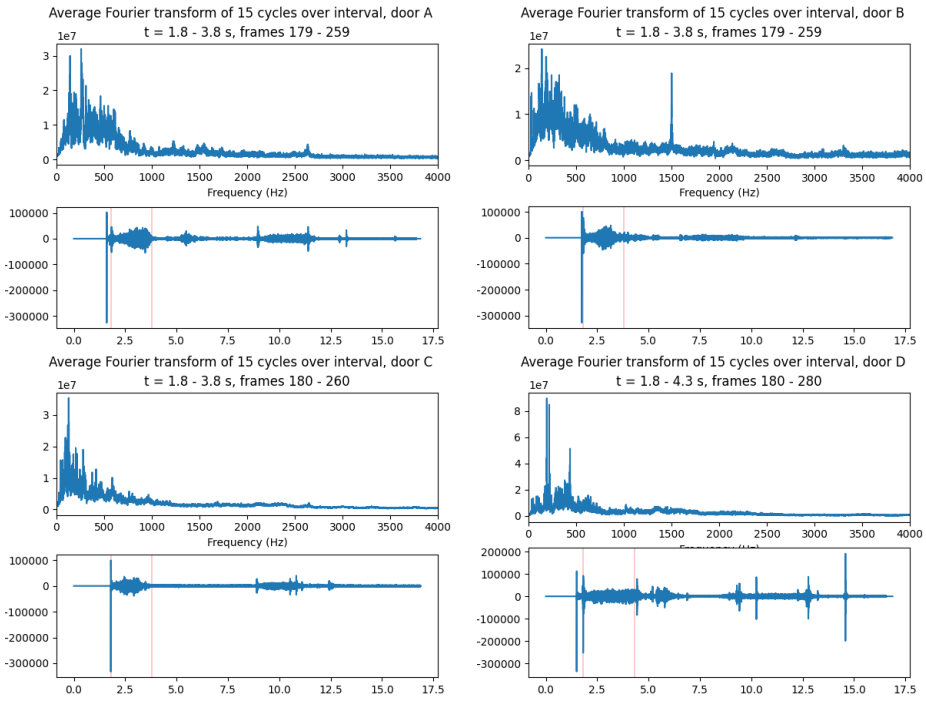


Figure 4.1: The mean Fourier transform of 15 opening cycles for the different doors. The area highlighted in red represents the section of the recording where the door opening occurs.

4.2.2 Spectrogram

Spectrograms, showing how a spectrum of frequencies change in a signal over time, were created for the recorded clips. As each door type gave similar results across the 15 recordings, only 5 spectrograms per door type are shown in figure 4.2. In the spectrogram it can clearly be seen where the opening and closing of the door occurs, but no specific frequencies stand out. Because of this the spectrograms were not used for developing the detection algorithms.

4.2.3 Signal energy

The energy of the signals were calculated, and the averages for each door type can be seen in figure 4.3, where the section of the opening sound is highlighted. It is noticeable here that large energy spikes occur outside of the interval. The largest of these is in door A, which can be attributed to the loud "click" of the door unlocking. Similar sounds occur as the other doors are beginning to open, which explains the spikes of lesser magnitude present in the other door types. However, within the interval there is a noticeable raised level of energy for a sustained amount of time. When the door is closing the signal

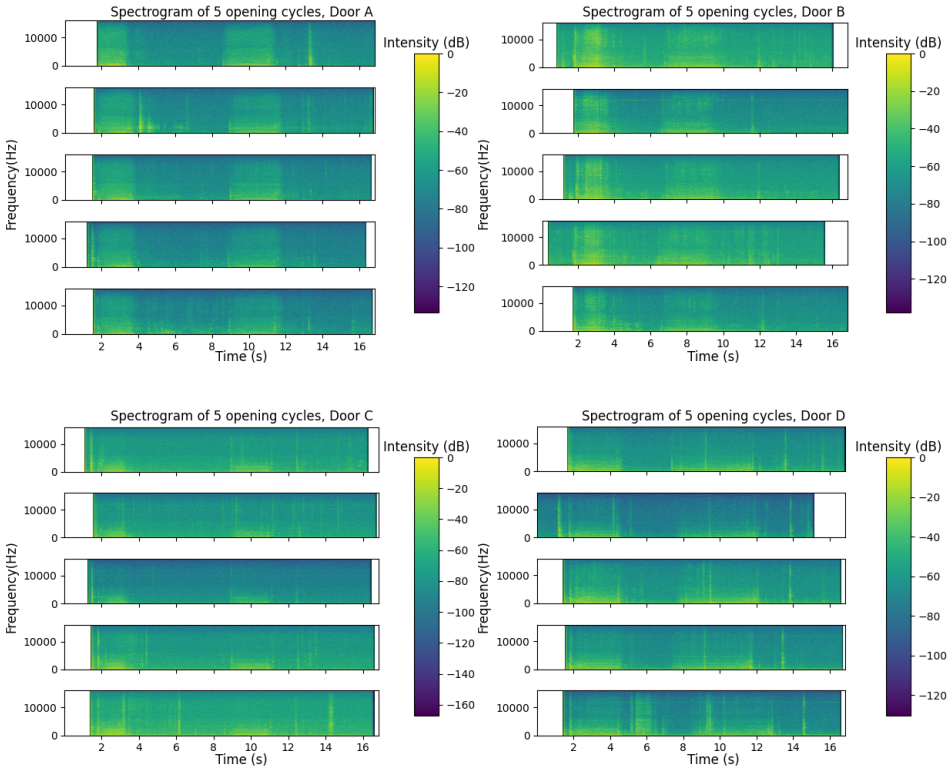


Figure 4.2: The spectrogram of 5 chosen opening cycles for each door type. The white space at the start and end of the clips are due to zero padding in order to synchronise the door openings.

energy is also raised for a length of time, but to a lower magnitude. This means an algorithm checking the energy level of a signal over time could be used to detect an opening cycle.

4.2.4 MFCC

Calculating the MFCC for a recording returned 13 coefficients for each frame calculated. In the `python_speech_features`[8] library used for the calculations, coefficient 0 was replaced by the energy present in the frame, and was therefore discarded. Between frames, the coefficients often changed drastically, but when averaged over a length of time their smoother changes over time could be seen. The average MFCCs for frames over the interval where the automatic door opens were calculated, and are displayed for every recorded cycle per door type, which are shown in figure 4.4.

These mean values of these coefficients per door type, as well as their standard deviations, can be seen in tables 4.1, and 4.2 respectively.

Table 4.1 shows that some coefficients differ greatly between door types, such as coefficient 1. However, there are several that are close in mean value, such as coefficients 5 or 6.

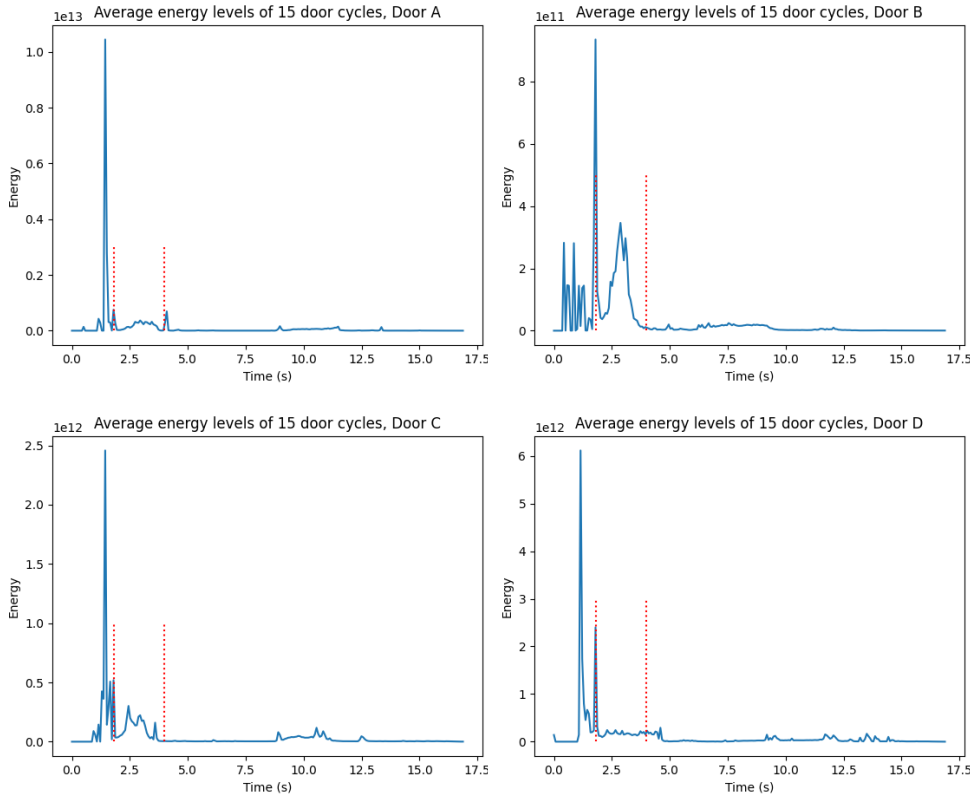


Figure 4.3: The average signal energy levels of the different doors. Highlighted in red is approximately the interval where the doors open.

Similarly, table 4.2 shows that most coefficients have fairly small standard deviations, which suggests that the mean values for each door type gives a fairly accurate impression of what values the coefficients have between individual door cycles. This suggests that an algorithm could be created where values for specific coefficients are detected. It is important to keep in mind that the values for individual coefficients move a lot over time, which implies false positives can spontaneously occur unless detection is done over a larger period of time.

4.3 Proposed algorithms

The findings from the pre-study gave way to two different algorithms, which are presented here. Because the detection is done in real time, the data is analysed in segments. Each segment consists of $N = 2048$ samples, which corresponds to 64 milliseconds of audio. Once an opening cycle is detected, the detector clears its buffers and detection is halted for 1.5 seconds. This is done in order to avoid several door cycles being detected during the same door opening.

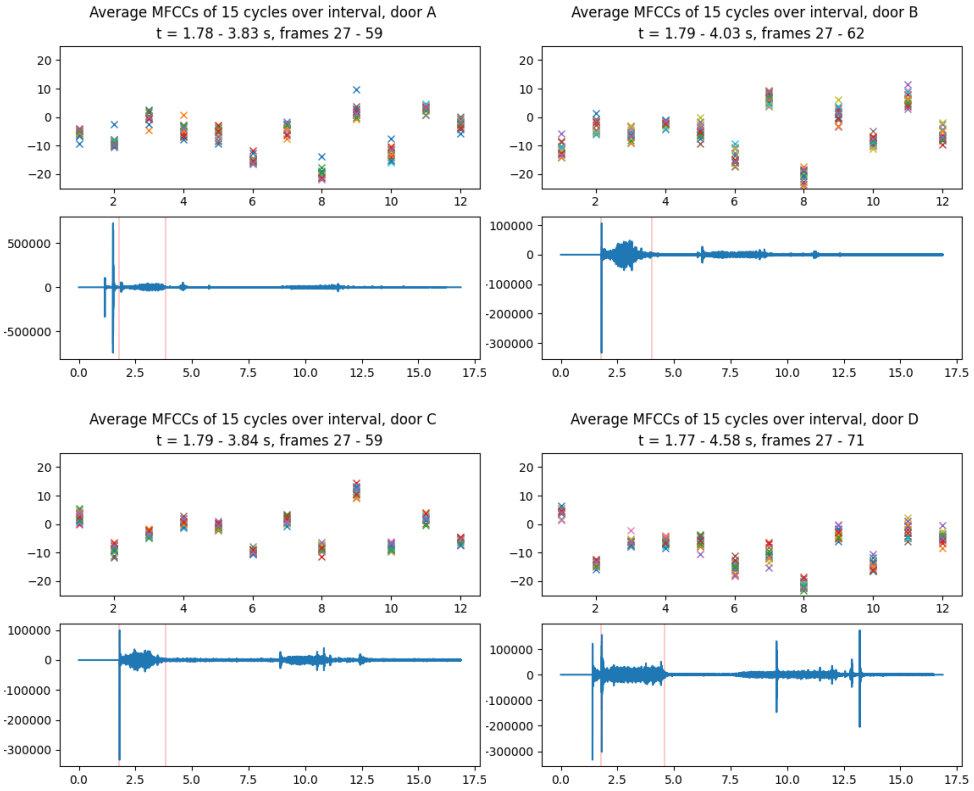


Figure 4.4: Mean MFCCs for the different doors. Each color represents an opening cycle, each having 12 coefficients. The interval highlighted with red vertical lines represents the section of the audio being analysed, and contains the sound the door makes during an opening.

4.3.1 Detection using signal energy

By analysing the signal energy for the recordings, it was concluded that opening cycles generally consist of a raised energy level for approximately one second. Because of this, the following detection algorithm was determined:

If for N segments in a row, the signal energy present in the segment $E_s = \sum_{n=1}^{N} x[n]^2$ exceeds the threshold E_t, an opening cycle is detected.

Based on the data, the amount of segments was set to $B = 20$, corresponding to about 1.3 seconds. The threshold was set to $E_t = 5 \cdot 10^{10}$ W/m^2

4.3.2 Detection using MFCC

By analysing tables 4.1 and 4.2, coefficients can be identified that are close in value between door types and have fairly low deviation between individual cycles. Using this information, reference values for the chosen coefficients can be decided, along with the expected range the values are expected to be within. These reference values and ranges

Coefficient	1	2	3	4	5	6	7	8	9	10	11	12
Door A	-5.9	-8.9	-0.5	-4.8	-6.1	-14.9	-3.7	-19.4	1.9	-12.9	2.4	-3.1
Door B	-10.9	-3.7	-6.0	-2.6	-5.2	-13.2	6.8	-19.8	0.5	-9.1	5.8	-6.8
Door C	2.7	-8.9	-3.9	0.2	-1.3	-9.7	1.4	-8.6	10.5	-9.1	1.1	-6.3
Door D	3.6	-12.7	-5.4	-5.0	-5.0	-13.2	-8.3	-19.3	-1.9	-13.1	-0.8	-3.7

Table 4.1: The mean MFCCs for 15 opening cycles of each door type.

Coefficient	1	2	3	4	5	6	7	8	9	10	11	12
Door A	1.3	1.8	1.6	2.1	1.9	1.2	1.4	1.8	2.2	2.0	1.1	1.6
Door B	2.1	2.1	1.7	1.1	2.2	2.3	1.8	1.8	2.3	1.5	2.0	2.0
Door C	1.5	1.5	1.0	1.1	0.9	0.6	1.2	1.1	1.3	1.1	1.2	1.0
Door D	1.3	1.2	1.3	1.3	1.6	1.8	2.0	1.2	1.7	1.8	2.3	1.6

Table 4.2: The standard deviation from the mean values shown in 4.1

were chosen and are displayed in table 4.3.

Coefficient	2	3	4	5	6	10	11	12
Reference value	-8	-3	-2.5	-4	-12	-11	2	-5
Range	± 5	± 3	± 3	± 3	± 3	± 3	± 3	± 3

Table 4.3: The reference values and ranges of selected MFCCs.

Using these references, the following detection algorithm was constructed:

For every segment, the moving mean MFCCs of the last P segments are calculated and compared to the selected reference values. If the value of a mean coefficient is within the given range of its reference value the coefficients **match**. If the amount of matches for a segment exceeds the tolerance T, a **segment match** has occurred. If, in the last B analysed segments, the amount of segment matches is greater than the detection limit d, an opening cycle is detected.

Based on the results of the pre-study, the parameters were set to $P = 30$, $B = 30$, $T = 6$, and $d = 10$.

Chapter 5

Result

The algorithms that were determined in the pre-study were tested on recorded data, as well as in real time. The results of this are presented in this section.

5.1 Detection using signal energy

The algorithm using signal energy, described in the previous section, was implemented in Python for analysing the pre-recorded data, as well as in C for the real-time detector.

5.1.1 Results on test data

The algorithm was initially tested on the recorded audio from the pre-study. In order to match the conditions of the real-time analysis, the signal was divided into segments of similar length to when processing audio in real time. The results are shown in table 5.1, which features a total detection rate of 93%.

Door type	Door A	Door B	Door C	Door D	Total
Cycles detected	15/15	12/15	14/15	15/15	56/60
False positives	1	0	0	0	1

Table 5.1: Results of using signal energy to detect opening cycles from pre-recorded data.

5.1.2 Real-time analysis

The algorithm was successfully implemented on the microcontroller. The algorithm was tested on 20 door cycles of each door type in real time, and the results are shown in table 5.2. The resulting total detection rate was 96%, with only a single false positive. The detector is unaffected by impulses of interference, such as the click from an unlocking door, as it does not pass the threshold for long enough. However, longer noise in close proximity to the detector will trigger the microcontroller and incorrectly detect an opening cycle, such as blowing into the microphone.

Door type	Door A	Door B	Door C	Door D	Total
Cycles detected	20/20	18/20	19/20	20/20	77/80
False positives	1	0	0	0	1

Table 5.2: Results of using signal energy to detect opening cycles in real time.

5.2 Detection using MFCC

The algorithm for detecting MFCC, which is described in the pre-study, was implemented for testing on pre-recorded data, as well as for real-time analysis. The implementation for the pre-recorded data was implemented in Python, where the MFCC was calculated the same way as in the pre-study. For the real-time implementation in C the MFCC was calculated using the CMSIS-DSP library,[3] which features functions for calculating MFCC.

5.2.1 Results on test data

The reference values for the MFCC coefficients from the pre-study were used for the Python algorithm. The results of this are shown in table 5.3. The total detection rate is 83%, although there were also many false positives present.

Door type	Door A	Door B	Door C	Door D	Total
Cycles detected	14/15	14/15	12/15	10/15	50/60
False positives	13	14	2	10	39

Table 5.3: Results of using MFCC to detect opening cycles from pre-recorded data.

A majority of the false positives present occur during the closing sound of the door, which, while not desirable, is understandable given the similar characteristics of the sound to the opening sound.

5.2.2 Real-time analysis

The MFCC-based detection algorithm was successfully implemented on the microcontroller. Due to differences in the libraries used when calculating the MFCCs there were slight differences in the coefficients calculated in real time in comparison to the ones from the pre-study. This meant that the reference values and ranges had to be adjusted to the ones seen in table 5.4.

Coefficient	2	3	4	5	6	10	11	12
Reference value	1	3	1.75	1	-1.5	-3.5	2	-1
Range	±3	±3	±2	±2	±3	±2	±3	±2

Table 5.4: The reference values and ranges of selected MFCCs, adjusted for the real-time analysis.

Table 5.5 shows the result of testing the real-time algorithm on 20 opening cycles per door type. The total detection rate was 55%. Using this algorithm for the detector

will very consistently detect opening cycles for doors A and D, and is more resistant to random noise such as blowing into the microphone. However, it often falsely detects an opening cycle during the closing of the door, as the sound is very similar to the opening sound. It also struggles with doors B and C much more than in the test results, which may be due to issues with the adjusted real-time parameters. Another problem is that loud noise close to the microphone during the opening of the door may interfere with the detector, causing a missed cycle.

Door type	Door A	Door B	Door C	Door D	Total
Cycles detected	20/20	6/20	0/20	18/20	44/80
False positives	14	3	0	12	29

Table 5.5: Results of using MFCC to detect opening cycles in real time.

5.3 Detection using both algorithms

A configuration was tested where the criteria of both algorithms had to be fulfilled in order for an opening cycle to be detected. As both algorithms already had been implemented both for analysis of pre-study data as well as for real-time detection, combining these was fairly trivial.

5.3.1 Results on test data

Attempting to detect opening cycles in the pre-recorded data with both algorithms produced the results seen in table 5.6. The total detection rate was 75%.

Door type	Door A	Door B	Door C	Door D	Total
Cycles detected	14/15	8/15	12/15	11/15	45/60
False positives	0	0	0	0	0

Table 5.6: Results of using both algorithms to detect opening cycles from pre-recorded data.

5.3.2 Real-time analysis

Testing both algorithms at the same time for 20 opening cycles of the 4 different door types yielded the results shown in table 5.7, with a total detection rate of 48%. For door B, both algorithms occasionally detected the same opening cycle, but because the signal energy criteria was only satisfied in the middle of the opening and the MFCC criteria only activated towards the end of the opening sound, the detections did not overlap, counting as a missed cycle.

Door type	Door A	Door B	Door C	Door D	Total
Cycles detected	20/20	0/20	0/20	18/20	38/80
False positives	0	0	0	0	0

Table 5.7: Results of using both algorithms to detect opening cycles in real time.

Chapter 6

Discussion

In this section, the results are discussed, along with potential improvements and alternative methods to consider for future research.

6.1 Comparison of algorithms

In an ideal scenario where only sound from the automatic door is present, the signal energy detection algorithm is very effective with a high detection rate and very few false positives for both pre-recorded data and real-time audio. Because the closing sound tends to be more quiet than the opening sound, the two sounds can be differentiated with this algorithm. It is also simpler to compute when compared to MFCC, only requiring a single operation for each sample. In practice, however, the algorithm is vunerable to false positives, as any sustained loud noise will indiscriminately be regarded as an opening cycle. ASSA Abloy may consider using this algorithm in environments where there is expected to be little interfering noise, or when minimal usage of computational resources is a priority.

The algorithm using MFCC on the pre-recorded data has a slightly lower detection rate than the energy-based method, but many more false positives. These are primarily detected during the closing of the door, which explains why most of the door types have a similar amount of detected cycles and false positives. There is a large discrepancy between the pre-recorded results and the real-time results, as doors B and C gave very poor results in real time. This may be due to the fact that the parameters had to be changed for the algorithm, and that further adjustments are required to improve the results. Doors B and C are very quiet and sound very different from doors A and D, which could explain the difference in result between the two groups. The algorithm may still occasionally trigger a false positive from external noise, but has a higher tolerance for prolonged noise than the signal energy algorithm. In the present state of the algorithm it will require either that the doors being detected are compatible with the algorithm like doors A and D, or that the parameters are adjusted to work for more door types in order for the algorithm to be viable. Performance must be kept in mind as well, since the MFCC transform is more complex than calculating the signal energy.

Using both algorithms simultaneously on the data from the pre-study has the lowest detection rate for both the pre-recorded data and the real-time analysis, which is reasonable given that both criteria need to be true at the same time. As this method uses the MFCC algorithm, the results were poor for doors B and C. When looking at doors A and

D, however, there was a high detection rate with no detected false positives. Using both algorithms combines the ability to differentiate the opening sound and the closing sound with the signal energy algorithm, and the greater resistance to outside noise from the MFCC algorithm. Assuming the MFCC parameters can be adjusted for a better result, this means that at the cost of a somewhat lower detection rate, using both algorithms could provide the most resistance to false positives. In most realistic scenarios this shows potential to become the method that achieves the best results. While using the two algorithms for detection has shown to be possible on a microcontroller, it has not been determined if it will work on the IoT gateway ASSA Abloy intends to use it on, as several other functions must be performed at the same time.

To summarize, the signal energy algorithm presently yields the best results in an ideal environment, but combining it with the MFCC algorithm shows greater promise for more practical scenarios.

6.2 Alternatives to sound analysis

Sound analysis for counting opening cycles has a couple of advantages. The hardware is very cheap, easy to install and can be used for other purposes, such as determining the health of an automatic door, which has been researched before.[12]

However, there are a number of problems that arise when performing sound analysis in this way. Sound analysis is inherently more prone to false positives and negatives, especially if subjected to external noise. This is further amplified by the fact that different kind of doors might emit different sound characteristics when opening and closing. Furthermore, in a usage scenario such as the one for ASSA Abloy, performing sound analysis on a microcontroller can be taxing on a system which already needs to perform other tasks. There may also be some security minded customers with concerns regarding including a microphone coupled with storage media and an internet modem on all their doors, as would be the case if using the IoT gateway.

If these issues outweigh the advantages to using sound analysis, it could be considered to use an alternative solution for detecting opening cycles. One option could be an accelerometer, which is a cheap sensor that can detect movement in three axis. This would be ideal for detecting movement by mounting a sensor on the door. Another option could be a Hall effect sensor, another cheap sensor which can be used to measure magnetic fields. Since an electric motor by nature generates a lot of magnetic interference when running, a Hall effect sensor could be placed near it. Both of these methods could easily interface with an IoT gateway, and would require less computational complexity than sound analysis as well.

6.3 Further research

As this report has attempted to create initial concepts for detecting opening cycles, improvements on the research can still be made, as well as answering other questions that were not addressed here.

Using the methods described in this report, it may be possible to get improved results by adjusting the parameters. This may especially be the case for the MFCC algorithm, as there were several parameters to adjust, such as the choice of reference coefficients and their values. Additionally, if further analysis of the MFCC results from the pre-study would indicate that some coefficients are of more importance to the characteristics of

the opening sound than others, accuracy may improve by applying a weight function on the results. There may also be possibilities to improve the results with different hardware. For example, if a setup could be constructed that blocks out external noise to the microphone, the energy-based algorithm would face less false positives in noisy environments. There may also be entirely different methods for detecting opening cycles that were overlooked during the pre-study which could produce different results.

An initial constraint for the project was that all doors started in a fully closed position before they opened, but in practice they sometimes only close partly before opening again. Testing for effectiveness in these scenarios still needs to be done, and may need further adjusting of parameters for it to work.

While attempts have been made to make the code fairly efficient, performance has not been the primary focus for the implemented solutions. As the real world usage will be on a microcontroller simultaneously performing other tasks, it may be of interest to examine how much processing power and active memory each method requires, and if this is compatible with the existing hardware and software used by ASSA Abloy. There are also possibilities to improve the performance of the algorithms. One example would be to use fixed point notiation for the MFCC calculations, which was not implemented but is described in the appendix. Another method of reducing computational complexity would be to lower the sampling frequency, as well as storing these values with lower resolution.

Because the opening sound of an automatic door can be differentiated from the closing sound, there could be an alternative solution that attempts to detect both before registering an opening cycle. A problem that might occur with this method is when the door is only closed halfway before being reactivated and opening again, which according to ASSA Abloy counts as a second door cycle. This would mean ASSA Abloy would have to redefine what counts as an opening cycle, which may not be in their interests.

A challenge with the solution was the fact that the parameters had to be set in order to detect opening cycles for a wide variety of doors. However, if an alternative solution calibrated parameters specifically for the door the gateway is connected to, the accuracy for each door might improve. Because of this, it may be of interest to research the possibility of configuring the detector for the door upon startup.

Chapter 7

Conclusion

Studying the characteristics of opening cycles from different types of automatic doors led to the development of two different algorithms: one using signal energy and one using MFCC. These were tested alongside testing detection using both algorithms at the same time.

Using signal energy for detecting opening cycles of automatic doors works well when there is no disturbing noise, but is prone to false positives in more realistic environments. Using MFCC for detection will often generate false positives when the door is closing, and it is difficult to configure parameters that are general enough for every type of door. Using both algorithms simultaneously could potentially be both general enough to work for several door types and be resistant to false positives, but this would require improving the real-time MFCC detection.

In order to improve the results, there can be further adjustments of the algorithm parameters. There may also be other characteristics of the opening sounds that could be used for detection of opening cycles. If ASSA Abloy are interested in increasing stability of the detector with further research, they may want to consider redefining an opening cycle to consist of both an opening and a closing of the door. Alternatively they could try using alternative sensors for detection, such as an accelerometer or a Hall effect sensor.

With the results of the study, it is suggested to ASSA Abloy to either use the energy-based algorithm or further improve the configuration using both algorithms. While the current solutions are not yet perfect, they have laid the groundwork for improved versions, which may in time provide robust and general detection of opening cycles for various types of automatic doors.

Bibliography

[1] Firoz Shah A. et al. "Speaker Independent Automatic Emotion Recognition from Speech: A Comparison of MFCCs and Discrete Wavelet Transforms". In: *2009 International Conference on Advances in Recent Technologies in Communication and Computing*. 2009, pp. 528–531. DOI: 10.1109/ARTCom.2009.231.

[2] Akilaa. *Digital biquadratic filter, Direct Form*. Creative Commons Attribution-Share Alike 3.0 Unported license. 2010. URL: https://commons.wikimedia.org/wiki/File:Biquad_filter_DF-I.svg.

[3] ARM Limited. *CMSIS DSP Software Library*. Version 1.14.4. 2023. URL: https://arm-software.github.io/CMSIS-DSP/latest/index.html.

[4] ARM Limited. *CMSIS-DSP Real FFT Functions*. Version 1.14.4. 2023. URL: https://arm-software.github.io/CMSIS-DSP/latest/group__RealFFT.html.

[5] Maria Sandsten Georg Lindgren Holger Rootzén. *Stationary Stochastic Processes for Scientists and Engineers*. CRC Press, 2014, p. 245. ISBN: 978-1-4665-8618-5. URL: https://www.maths.lu.se/forskning/boecker/stationary-stochastic-processes-for-scientists-and-engineers/.

[6] Audrey F. Harvey. "DMA Fundamentals on Various PC Platforms". In: 1994. URL: https://cires1.colorado.edu/jimenez-group/QAMSResources/Docs/DMAFundamentals.pdf.

[7] James Lyons. *Mel Frequency Cepstral Coefficient (MFCC) tutorial*. URL: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/.

[8] James Lyons. *python_speech_features*. Version 0.6.1. 2020. URL: https://doi.org/10.5281/zenodo.3607820.

[9] Mathworks. *butter - MATLAB*. Accessed on June 5, 2023. 2023. URL: https://se.mathworks.com/help/signal/ref/butter.html.

[10] Zied Mnasri et al. "Anomalous sound event detection: A survey of machine learning based methods and applications". In: *Multimedia Tools and Applications*. Vol. 81. 2022, pp. 5537–5586. DOI: 10.1007/s11042-021-11817-9.

[11] Douglas O'Shaughnessy. *Speech Communication: Human and Machine*. Addison-Wesley series in electrical engineering. Addison-Wesley Publishing Company,

1987, p. 150. ISBN: 9780201165203. URL: https://books.google.se/books?id=mHFQAAAAMAAJ.

[12] Olof Englund. *Failure prediction for mechanical doors using cheap sound analysis*. eng. Student Paper. 2021. URL: https://lup.lub.lu.se/student-papers/search/publication/9074394.

[13] Kuldip K. Paliwal. "Decorrelated and liftered filter-bank energies for robust speech recognition". In: *Proc. 6th European Conference on Speech Communication and Technology (Eurospeech 1999)*. 1999, pp. 85–88. DOI: 10.21437/Eurospeech.1999-25. URL: https://maxwell.ict.griffith.edu.au/spl/publications/papers/euro99_kkp_fbe.pdf.

[14] Purple-bobby. *Digital Biquad Direct Form 2 Transformed*. Creative Commons Attribution-Share Alike 4.0 International licence. 2018. URL: https://commons.wikimedia.org/wiki/File:Digital_Biquad_Direct_Form_2_Transformed.svg.

[15] Nigel Redmon. *earlevel.com, Biquads*. Accessed on 31 May, 2023. 2003. URL: https://www.earlevel.com/main/2003/02/28/biquads/.

[16] Asif Salekin et al. "A Real-Time Audio Monitoring Framework with Limited Data for Constrained Devices". In: *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. 2019, pp. 98–105. DOI: 10.1109/DCOSS.2019.00036.

[17] Siemens. *Introduction to Filters: FIR versus IIR*. Accessed on June 5, 2023. 2023. URL: https://community.sw.siemens.com/s/article/introduction-to-filters-fir-versus-iir.

[18] The SciPy community. *scipy.fftpack.dct — SciPy v1.10.1 Reference Guide*. Accessed on June 5, 2023. 2023. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.fftpack.dct.html.

[19] The SciPy community. *scipy.signal.butter — SciPy v1.10.1 Reference Guide*. Accessed on June 5, 2023. 2023. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.butter.html.

[20] The SciPy community. *scipy.signal.windows.hamming — SciPy v1.10.1 Reference Guide*. Accessed on June 5, 2023. 2023. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.windows.hamming.html.

[21] Fayçal Ykhlef et al. "Real-Time Detection of Impulsive Sounds for Audio Surveillance Systems". In: *National Study Day on Research on Computer Sciences*. 2019.

Appendix A

Fixed point notation

This project has utilized floating point operations for all calculations. Ideally floating point operations are not used on microcontrollers, which is why the CMSIS-DSP C library used in the project includes alternatives to every floating point which utilize fixed point notation. For example, for every `arm_mfcc_f32()` or `arm_mfcc_f16()` function there also exists a `arm_mfcc_q31()` or a `arm_mfcc_q15()` function, depending on the desired resolution of the computation.

Although this wasn't used in the project, future iterations of the algorithm may want to use this when when moving away from floating point representation of numbers to increase performance.

A.1 Two's complement

To represent negative numbers in binary, two's complement is the most common method, as is visualised in figure A.1. This method is almost universally used to represent negative numbers on all computers.

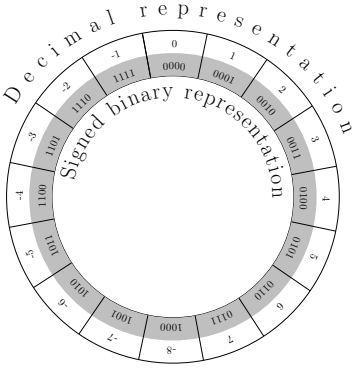


Figure A.1: Two's complement representation

For example, adding the numbers 5 and -4 would equate to adding the numbers `0101` and `1100` in two's complement for 4 bit numbers. This would add up to be `10001`, a 5 bit number. The most significant bit is disregarded, which leads to the correct result `0001`=1.

A.2 Fixed point representation

Fixed point numbers are usually represented with the `Qm.n` notation, where `m` is the integer part and `n` is the fractional part and is stored in memory as shown in figure A.2

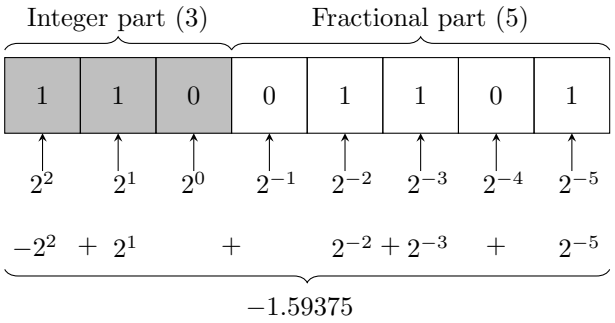


Figure A.2: Example of a number represented with Q3.5 notation with two's complement

For the CMSIS-DSP library all fixed point numbers are represented either as `1.15` or `1.31`, being only capable of representing numbers between 1 and -1. The limitations that arise due to only representing numbers in this range is mitigated with techniques such as scale factors.

Printed by Tryckeriet i E-huset, Lund 2023

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2023-944
http://www.eit.lth.se