Tel-Aviv University

Faculty of Engineering

School of Electrical Engineering

אוניברסיטת תל-אביב

הפקולטה להנדסה

בי"ס להנדסת חשמל

Glass Breakage Detector

Project Number 16-1-1-1126

Final Report

Students:

Assaf Bar Ness	303048698
Assaf Ben Yishay	302747621

Instructor:

Arkadi Rafalovich

Table of Contents

1 <u>Abstract</u>

Classification of a glass breakage event is an alarm-system related problem.
This problem can be solved using different methods, each with its own unique flaws and benefits.
The goal of this project is to solve this problem with the following constraints:

- Minimal false detection probability.
- Minimal false alarm probability.
- General solution as possible, regardless of the system physical position.
- Zero calibration done by user.
- Low as possible current consumption in order to achieve maximal run time without replacing the battery.
- Small and compact dimensions.

In today's industry, ARM architecture microcontrollers are widely used for variety of applications. These microcontroller main characteristics are durability, high performance, reliability and low price. In addition, many of the vendors that manufactures these microcontrollers add floating point hardware for better support of DSP functions.

This project core component is ARM microcontroller by ST electronics.
A high level block diagram of the suggested solution is shown below:

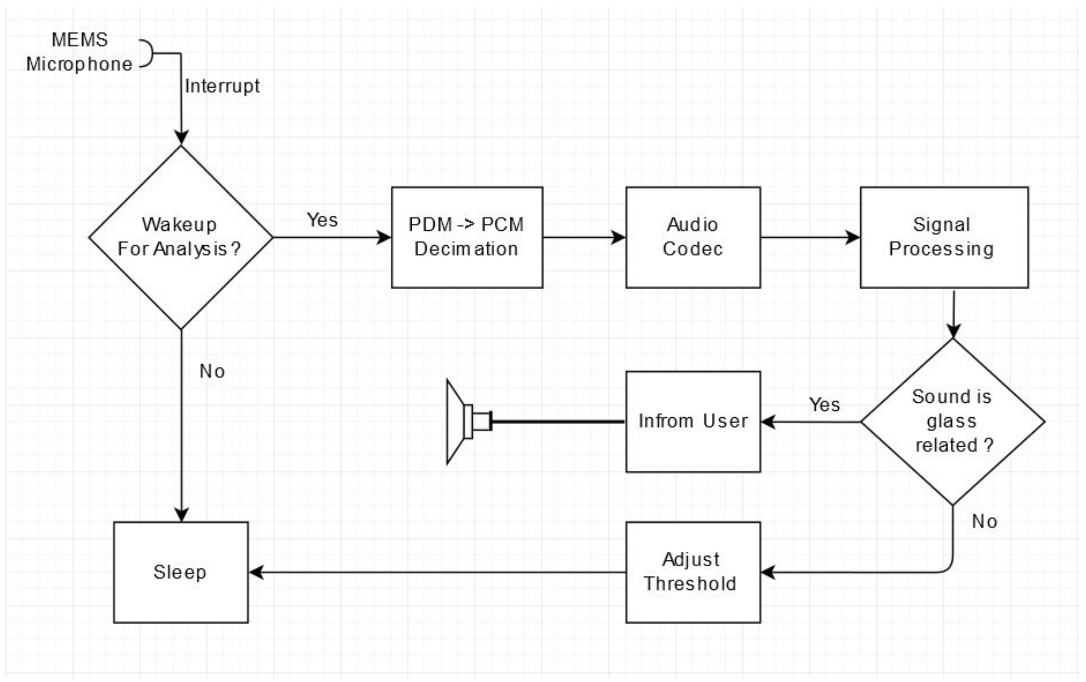


Figure 1 – Suggested Solution Block Diagram

2 Introduction

The projects main challenge is to recognize a glass breakage event in real time. It is important to notice that the system is triggered all the time by false events and that the probability of a true event is extremely small. This is why we must demand that the miss detect and false alarm probabilities will be as small as possible.

This problem is part of a wider family of problems dealing with signal type classification (when the signal is partly random) and their translation to data which other layers of software can handle. Similar problems are speech recognition, biological and physical signals classification and so on.
Thus, the approach for solving this problem is not unique to glass breakage event but could be implemented for solving other problems as well, although most application processing power will not be embedded microcontroller.
Some of the solutions that are widely used today are hidden markov model (HMM), spectrogram image analysis, general signal processing algorithms and pattern recognition.

The algorithm we chose to implement in order to solve this problem is logistic regression, with additional layers of signal processing and sound localization.
Logistic regression is a regression model where the dependent variable is categorical (in our case is binary – true or false). The regression coefficients are estimated using maximum likelihood and are found iteratively. The independent variables are chosen by us and must represent the glass breakage sound properly in order to ensure reliable classification.
Thus, there are many tradeoffs between computation power of the system in real time and the resolution and accuracy of these features.
When using features which models the event properly, it is possible to distinguish between daily sound events (false alarm) to true events.

Similar work has been done by Texas Instruments [1] using the MSP430 8-bit microcontroller and by Cypress [2] (CY8C22xxx).
TI's and Cypress solution are based on single or two recordings of glass breakage events. Thus these solutions are too basic for general use. The solution we are suggesting for this project is more general and more fundamental and can be used for events that the algorithm did not learn and faced before.

3 Theoretical Background

The goal of this project is to classify glass breakage sound from daily house/building noises. Because glass breakage sound is not deterministic and vary from place to place and even from time to time in the exact same environment conditions, the classification is rather hard and could not be always accurate.
In order to reduce false-alarm and miss-detect probabilities, it is important to analyze the statistical characteristics of such sound and use them in the classification procedure. It is also possible to improve the probabilities by using another independent test, such as direction of the incoming sound.

3.1 Spectral Analysis

One of the options to classify a random signal is by spectral analysis. There are many ways to estimate a random process spectrum.
In order to understand the characteristics of this kind of sound, four spectrum estimators of an example signal are shown in Figure 1.
Each estimator was tested on several different sections of the sound. It is very clear from the plots that the spectrum is not constant in time. This will be an important aspect of the detection algorithm.

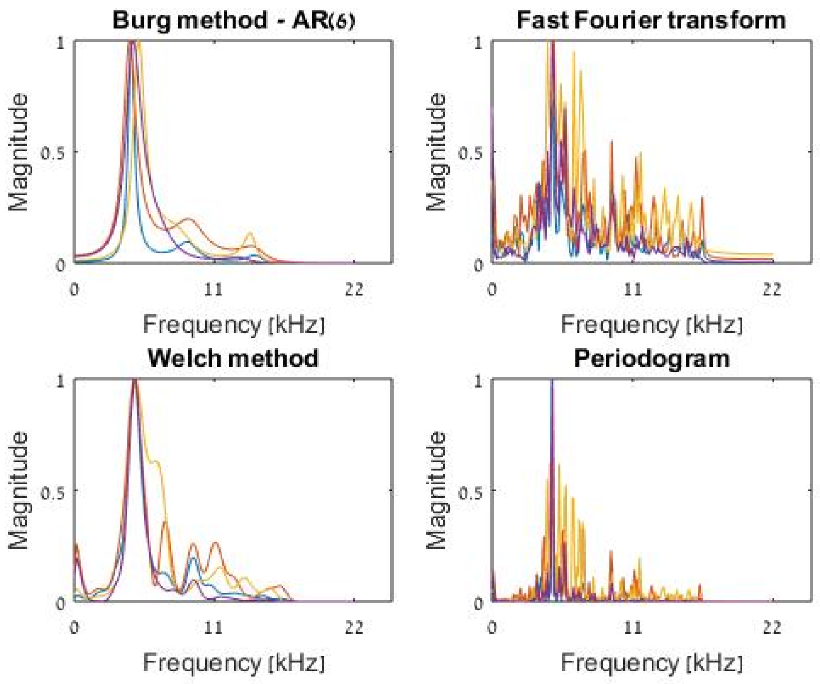


Figure 2 – Comparison of four spectrum estimators

This insight of the glass breakage event characteristic tells us that we need to use a method which not relies on the stationary of the signal, because glass breakage signal is definitely not stationary.

3.2 Thud Identification

Another important feature in glass breakage sound is the initial impact tone. This event is characterized by a low frequency tone (typically 300-1000 Hz) which can be detected easily by examining the first few hundreds of samples after the signal's energy cross a pre-defined threshold.

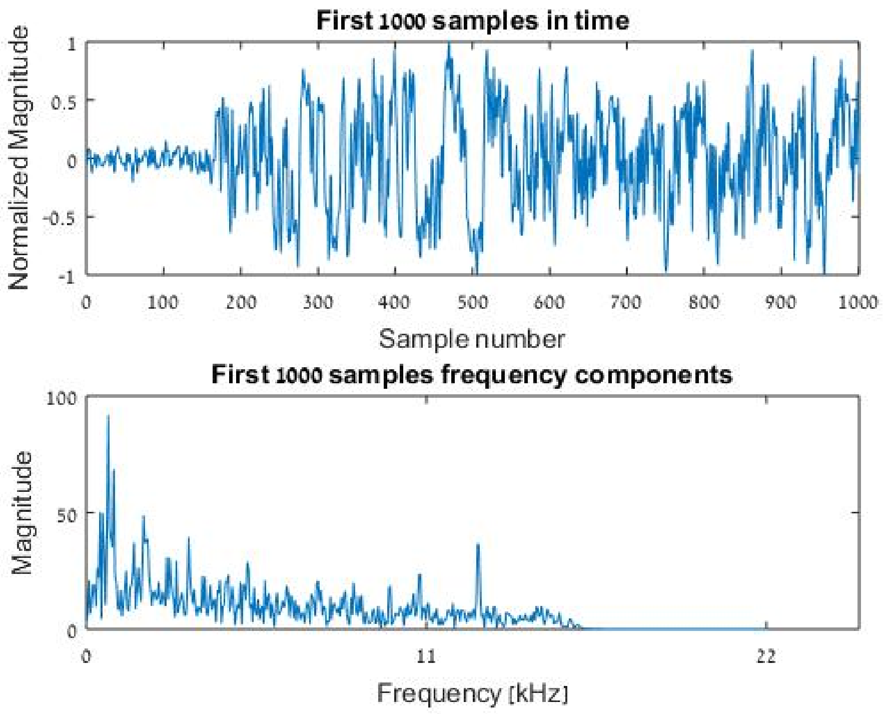


Figure 3 – A typical glass breakage process first 1000 samples, in time and frequency domain

It can be seen from the plots that at the beginning of the process, most of the energy is centered at the lower frequencies. This is not the case for greater samples indexes.
A simple and fast way of detecting a single tone is using the Goertzel algorithm, which is in fact a simple digital filter.
Another method is simply use a BPF or LPF with appropriate bandwidth.

3.3 Sound Localization

Sound localization refers to the ability to identify the location or origin of a detected sound in direction and distance. Such ability is particularly useful for glass breakage detection. It is possible to determine the direction of recorded sound by using array of microphones [4] (minimum two). By calculating the cross correlation between two recorded signals, and by knowing the speed of sound – the signals origin can be extracted.
For example, if we have two microphones in 0.5m distance and the sampling frequency is 44kHz, the same signal will be recorded in both microphones with a phase shift of 64 samples – in case that the direction is orthogonal to the microphones.
An efficient numerical computation of cross correlation can be done in frequency domain:

$$f \star g = \mathcal{F}^{-1}\{\mathcal{F}\{f\}^* \cdot \mathcal{F}\{g\}\}$$

Where $\mathcal{F}$ is the Fourier transform and $\star$ denotes the cross correlation operator.

If the signals length is relatively small, it is more efficient to calculate the cross correlation directly:

$$(f \star g)[n] = \sum_{m=-l}^{l} f[m] \cdot g[m+n]$$

Where l is the maximal expected delay between the signals.

3.4 Conclusion

Classifying a breaking glass sound cannot be done using a single test. In order to achieve maximum performance it is necessary to apply several independent tests, each giving a different information about the signal.

4 Technology Review

4.1 Microphones

The first device to interact with the recorded signal is a microphone.
ST offers different types of MEMS microphones which targeted to embedded applications. These microphones are small, reliable and can be integrated into any system which supports SPI, I2S or DFSDM.

A MEMS microphone is basically a silicon capacitor. The capacitor consists of two silicon plated. One plate is fixed while the other one is movable. Small holes in the movable plate allow the air pressure to change the capacitance of the capacitor. These small changes are PDM modulated by integrated ASIC.

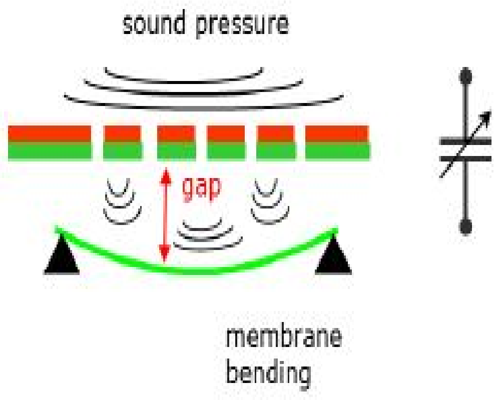


Figure 4 – Capacitance change principle

In the STM32L product line, DFSDM (Digital Filter for Sigma Delta Modulator) is used [5]. A clock signal is generated by the DFSDM peripheral and given in input to the microphone. The microphone provides a PDM serial output data which is processed by the DFSDM directly into PCM code. The data is then buffered and stored using DMA.

There are number of important parameters to be considered when using MEMS microphone[3]:

- Sensitivity – the sensitivity is the electrical signal at the microphone output to a given acoustic pressure. For digital microphones the sensitivity is expressed in dBFS. Level of 0 dBFS means that the signal is in its maximum digital level.
  The sensitivity of MP34DT01-M is -26 dBFS, meaning that the microphone can sense a signal that is 5% of the maximum level.
  This value should not be confused with dynamic range.
- Directionality – The sensitivity change due to different position of the sounds source. ST MEMS microphones are all omnidirectional, which means that the sensitivity is not affected by the sound's position.
- SNR – the ratio between a given reference signal to the amount of residual noise at the microphone output. The SNR of MP34DT01-M is 61 dB.
- Dynamic Range – The difference between the minimum and maximum signal that the microphone is able to generate as output, measured in pascal. For the MP34DT01-M, the ratio between the maximum and minimum signal is 120 dBSPL.
- Frequency response - the transfer function of the microphone

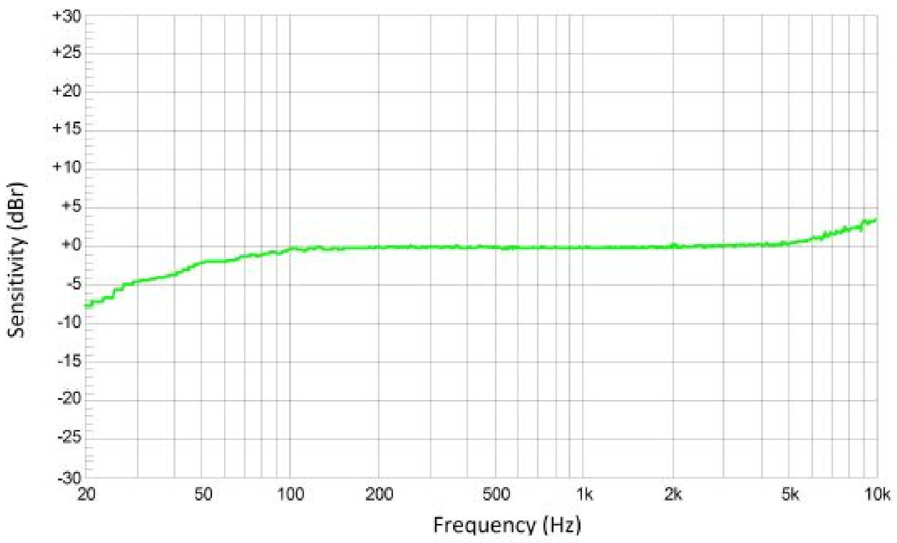
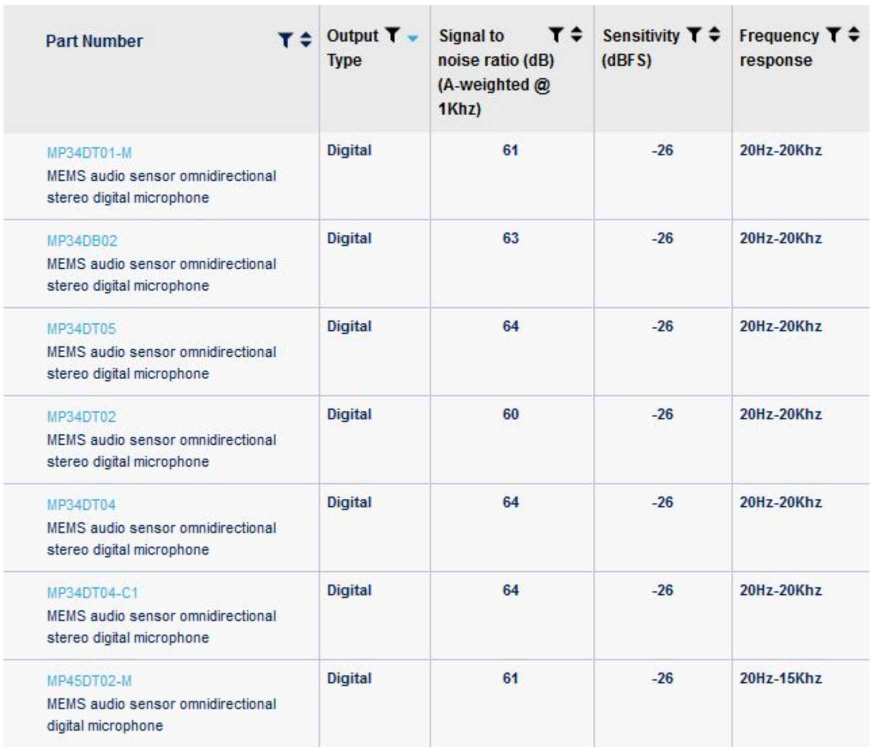


Figure 5 – Typical frequency response of the MP34DT01-M

A comparison between ST digital MEMS microphones is shown in table 1.



Part Number	Output Type	Signal to noise ratio (dB) (A-weighted @ 1Khz)	Sensitivity (dBFS)	Frequency response
MP34DT01-M MEMS audio sensor omnidirectional stereo digital microphone	Digital	61	-26	20Hz-20Khz
MP34DB02 MEMS audio sensor omnidirectional stereo digital microphone	Digital	63	-26	20Hz-20Khz
MP34DT05 MEMS audio sensor omnidirectional stereo digital microphone	Digital	64	-26	20Hz-20Khz
MP34DT02 MEMS audio sensor omnidirectional stereo digital microphone	Digital	60	-26	20Hz-20Khz
MP34DT04 MEMS audio sensor omnidirectional stereo digital microphone	Digital	64	-26	20Hz-20Khz
MP34DT04-C1 MEMS audio sensor omnidirectional stereo digital microphone	Digital	64	-26	20Hz-20Khz
MP45DT02-M MEMS audio sensor omnidirectional digital microphone	Digital	61	-26	20Hz-15Khz

Table 1 –Digital MEMS Microphones Comparison

It can be seen from the table that all microphones are designed to work in the same frequency ranges. The difference between the microphones frequency response is the sensitivity in the higher frequencies. In the range 0-10kHz all microphones shows flat response.
Since most of the energy of our target sound is in the lower frequency range of 0-10kHz, the sensitivity in the higher range will have minor influence on the classification algorithm decision. We expect that migrating from one microphone to another will require only thresholds update.

4.2 Microcontrollers

In order to process the sound acquired by the microphone, we will need a fast and efficient processor. We would also like to minimize the power consumption of such microcontroller so it will be suitable for low power applications.
The STM32L4 series of ultra-low power MCU's from ST are ARM Cortex M4 processors which characterized by high-performance, various peripherals, FPU, DSP instruction and of course ultra-low power.
We will develop our application on the STM32L476RG.

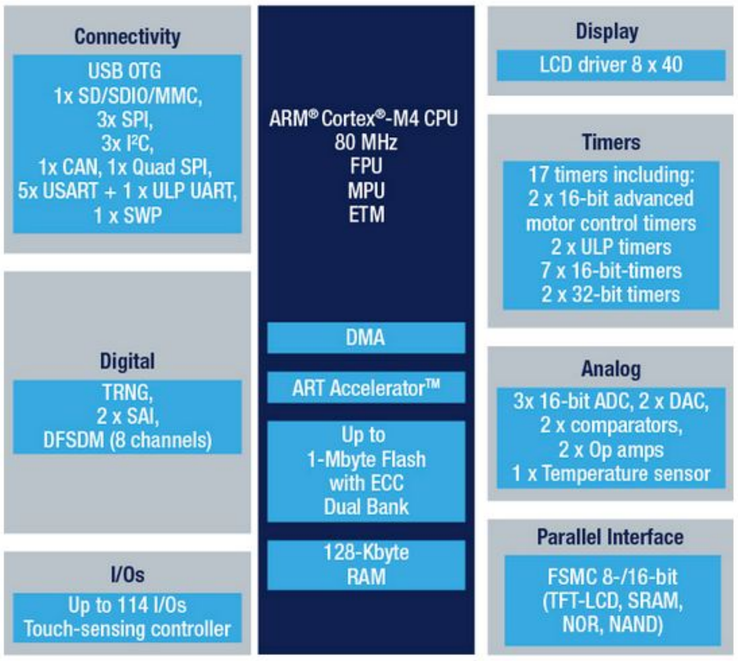


Figure 6 – STM32L476 Circuit Diagram

Some of the ST32L476 features[7]:

- 1.71V to 3.6V power supply
- 1.4 uA current in stop mode with RTC
- 100 uA/MHz run mode
- Up to 1MB Flash
- Rich analog peripherals – low power comparators, operational amplifiers, DAC
- Up to four DFSDM channels
- Up to 80 MHz clock frequency

5 Suggest Solution and Simulation

5.1 Problem Description

The goal of the projects algorithm is to detect glass breakage event. The main difficulties of detecting this type of event are:

- The event characteristic are stochastic by nature, so it is necessary to use probabilistic approach. Thus, learning a specific event of relying on specific characteristic is not enough.
- The only input of the microcontroller is the microphone, so the algorithm must rely only on the spectral characteristics of the event.
- Background noise could be present.

5.2 Solution

There are several approaches for solving this problem, which differs in the need of the following resources:

- Prior statistical information.
- Available computation power (e.g. solution complexity).
- Raw data quality.

Since our solution is targeted as low power and due to the lack of rich prior statistical information, our main considerations are ease of implementation and simplicity.
The chosen algorithm is logistic regression binary classifier.
We would also use sound localization technics in order to evaluate the event location and thus minimize false alarm when the event location is in the opposite direction of the glass.

Logistic Regression

Logistic regression is a regression model where the dependent variable is categorical. If there are only two categories the model is called binary.
The binary logistic model is used to estimate the probability of a binary response based on one or more independent variables.
In our case, the categorical variable is "Glass breakage occurred" and "Glass breakage did not occurred".

In a regression model, we approximate an event as a linear function of its characteristics (denoted $\{x_1, x_2, \dots, x_n\}$) . The predicted $\hat{y}(\underline{x})$ value is modeled:

$$\hat{y}(\underline{x}) = \theta_0 + \theta_1 x_1 + \cdots + \theta_n x_n$$

Where the θ_i's are the problem parameters that we're trying to optimize.

The probability that the event took place is given by:

$$g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

And for a binary classifier with two hypothesis, the decision rule is:

$$Y = \begin{cases} 1 & g(\theta^T x) \geq 0.5 \\ 0 & g(\theta^T x) < 0.5 \end{cases} = \begin{cases} 1 & \theta^T x \geq 0 \\ 0 & \theta^T x < 0 \end{cases}$$

In order to find the optimal parameters $\{\theta_i\}_{i=1}^{n}$, we define the following cost function:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left[-y^i \log\left(g(\underline{x}^i)\right) - (1 - y^i) \log\left(1 - g(\underline{x}^i)\right) \right]$$

Where x^i represent the i'*th* set of features of the i'*th* event and y^i indicates the i'*th* event type (e.g. true or false).

After extracting the features from the data, it is possible to approximate the parameters using numeric software like MATLAB.

<u>Example</u>:

for a binary classification problem with a single feature, the predicted value is given by:

$$\hat{y}(\underline{x}) = \theta_0 + \theta_1 x_1 + \theta_2 x_2$$

The decision boundary is $\hat{y}(\underline{x}) = 0$, or :

$$\theta_0 + \theta_1 x_1 + \theta_2 x_2 = 0 \rightarrow x_2 = -\frac{1}{\theta_2}(\theta_0 + \theta_1 x_1)$$

Plotting the data and decision boundary for an example dataset:

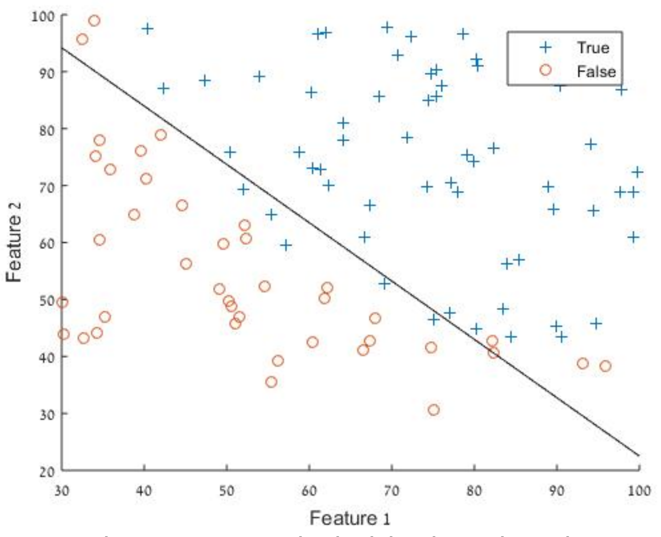


Figure 7 – Example decision boundary plot

Important notes:

- The decision rule does not guaranty 100% success.
- This example shows a simple case where there are only two features. In practice the number of features is larger.
- When there are more than two features, the decision boundary is a hyper plane.
- If the data is not linearly spreadable it is possible to use nonlinear characteristics.

Sound Localization

When using array of microphones, it is possible to evaluate the sound source location trough the different microphones cross correlation.
Suppose we have two microphones mounted on the same plane with distance of 4 cm, sampling the environment sounds with sampling rate of 32 kHz. Since the speed of sound is 340 m/s, a sound wave coming from the same axis as the microphones will be recorded by the two microphones with delay of:

$$Delay = Fs \cdot \frac{d}{v_{sound}} = 32000 \cdot \frac{0.04}{340} = 3.76$$

In this case, the delay will be approximately 3-4 samples.
It is possible to achieve this delay in low noise environment and when the microphones are not very close to a wall. If longer delay or higher angle resolution is needed, one can increase the sampling rate or the microphones distance.
Similar method is to calculate the RMS value of a short interval. The RMS voltage of the closer microphone will always be slightly higher.

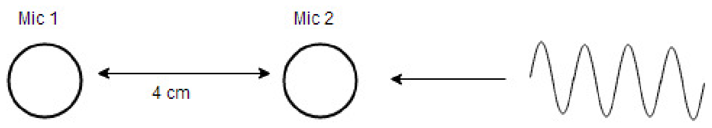


Figure 8 – Sound localization illustration

A recordings of two scenarios in which the sound source location is left and right is shown in the following figures:

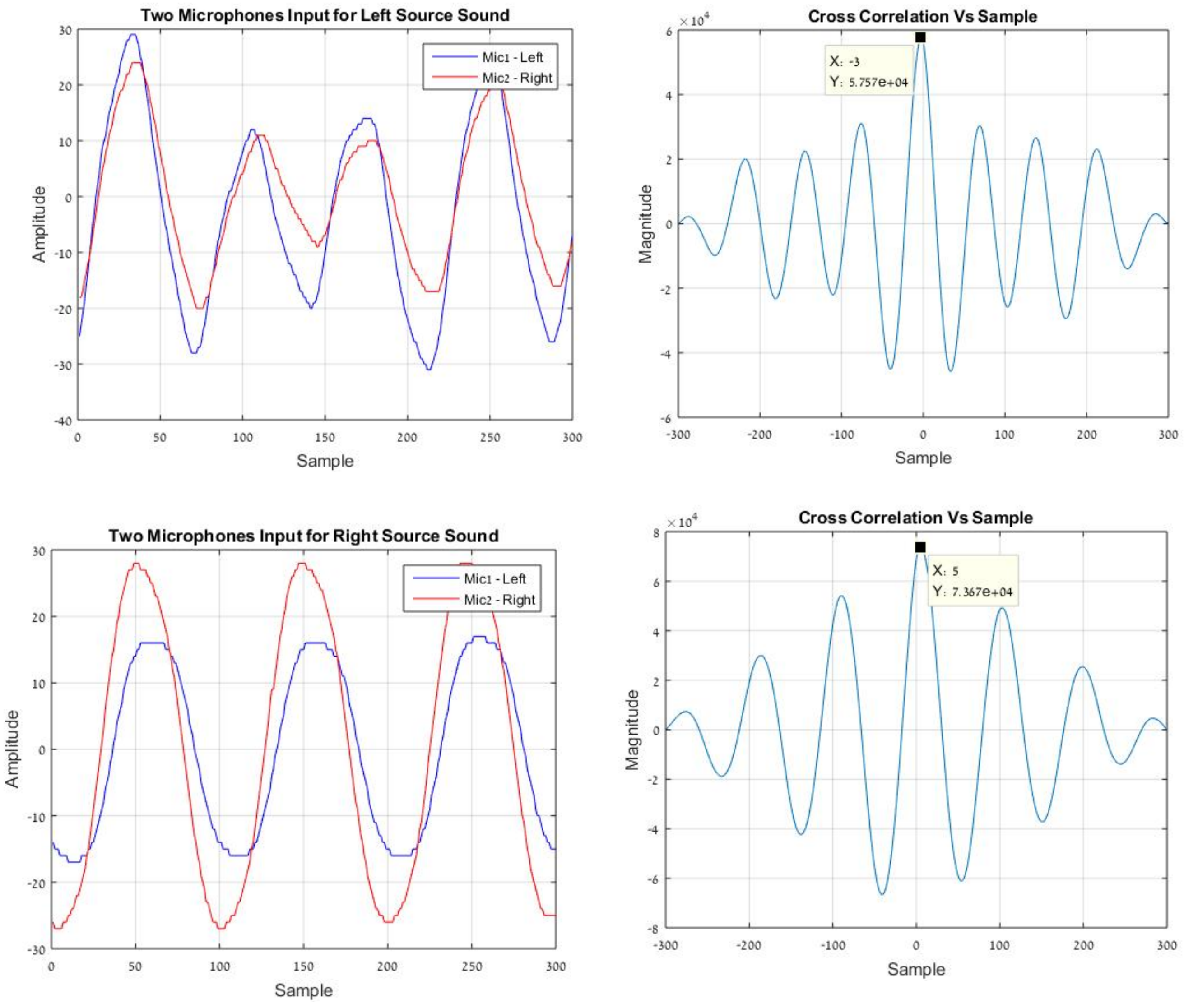


Figure 9 – Input signals and their cross correlation

Plots summary:

Event Location	Cross Correlation Maximum Value
Left	-3
Right	5

We can use the following decision rule:

$$location = \begin{cases} Right & \max(R_{xy}) > 2 \\ Left & \max(R_{xy}) < -2 \\ Unknown & Otherwise \end{cases}$$

In conclusion, we have seen that it is possible to determine a sound wave source without increasing the sampling frequency. This method can be useful as additional test to estimate if a sound event is glass breakage event – but only if the glass is located in specific part of the room.

5.3 Glass Breakage Features

The features extraction process is illustrated in the next figure:

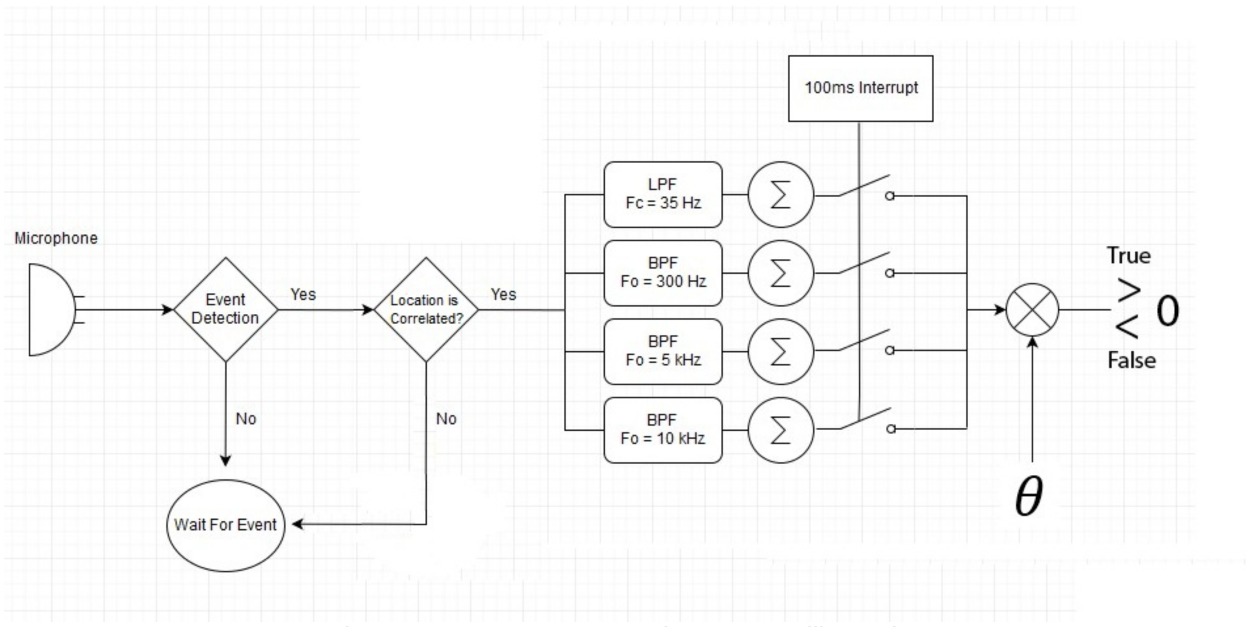


Figure 10 – Features extraction process illustration

Brief description:
Each audio packet, which consist of 32 samples is processed in order to detect new event. In case that the output of the detection unit is positive, a time window of 2 seconds is started. In that window, the data stream from the microphone is passed through 4 filters. Every 100 milliseconds (or frame), the mean sum of the absolute square output value of each filter is computed and stored in a dedicated array - $A_n[j]$ where n is the filter number and j is the frame. When the 2 seconds time window is finished, the four arrays (one for each filter) will be filled with 20 values (1 value for each frame). The event feature vector will be:

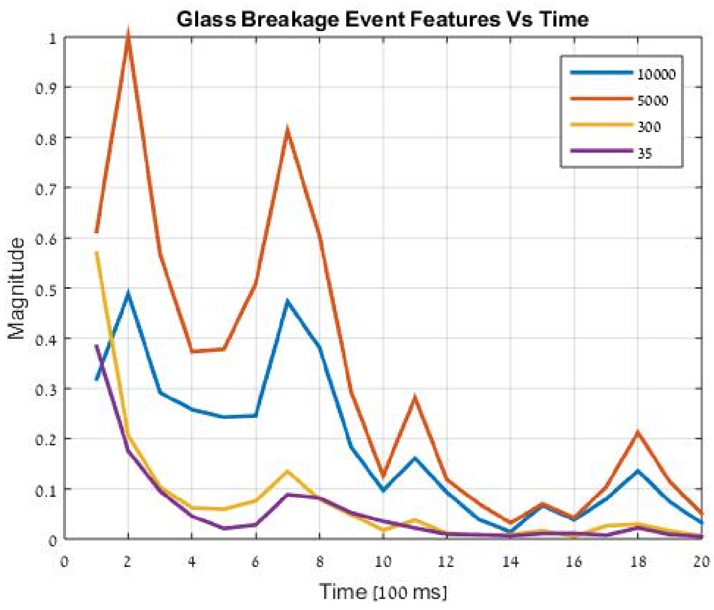
$$\underline{x} = [Filter1_{data} - Filter2_{data} - Filter3_{data} - Filter4\_data]$$

And the classification rule is:

$$\hat{Y} = \begin{cases} 1 & \underline{\theta}^T \underline{x} \geq 0 \\ 0 & \underline{\theta}^T \underline{x} < 0 \end{cases}$$

Where θ is the parameters vector.

For illustration purpose, a plot of the features are shown for true and false events:



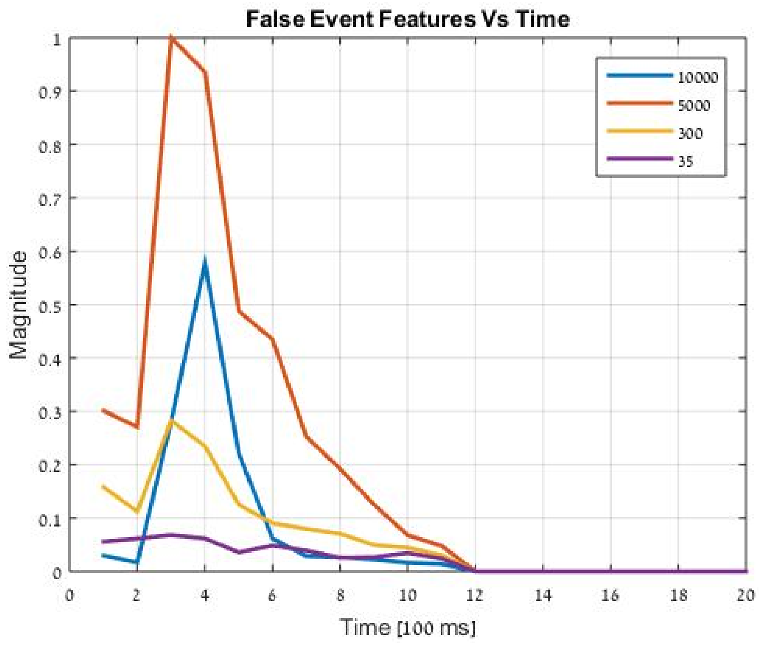


Figure 11 – True/False events features

It is important to notice that the various constants of the algorithm are not optimal in the analytic sense. These constants where chosen due to practical reasons – the MCU stack memory, system clock frequency, rate of spectral changes during typical sound events, etc. Thus, it is possible that another set of constant with better detection performance exist – with major tradeoffs in the MCU run time and power consumption, which we want to minimize as possible.

5.4 Glass Breakage Event Analysis

In order to achieve maximum accuracy in the event detection procedure, several glass breakage events were examined by their spectrogram:

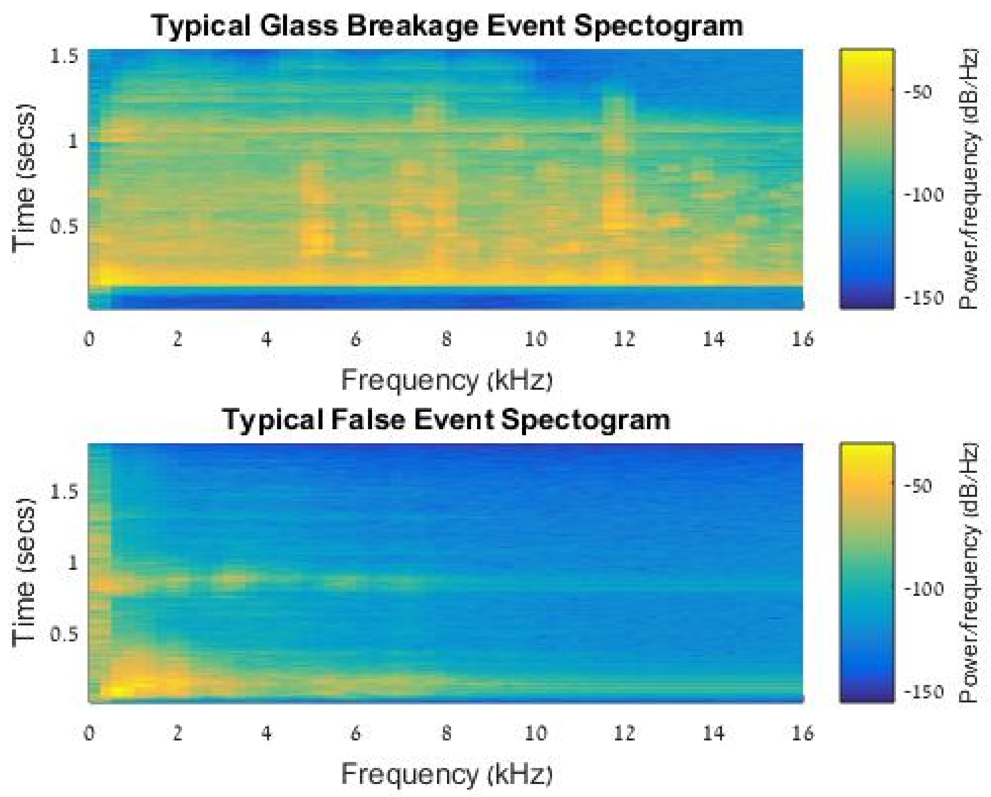


Figure 12 – True/False event spectrogram

From the plots we can see that the energy of a glass breakage event is spread uniformly in all frequencies were in some short time intervals or narrow frequencies bands the power density is higher.
In contrast to a glass breakage, a false event is usually shorter in time and the power is centered in finite number of frequencies.
For those reasons, the BPF filters were designed to pass only several bands which we believe characterize a typical glass breakage event (see software implementation).

5.5 Learning Procedure

The learning process is based on a dataset which contain 35 recordings. 20 records of breaking glass and 15 other records of similar daily life sounds – closing drawer, dock knocking, bell ringing, barking dog and etc.
The filtering procedure was applied to every recording, results the dataset features matrix. In order to avoid over-fitting, the matrix rows were shuffled and only 70% of them used as input for the fitting method.
The fitting method we have used is MATLAB's *"fminunc"*, which is a numeric method for finding a local minimum of a function of several variables using iterative gradient descent method.
After finding the optimal vector $\{\theta_i\}_{i=1}^{n}$ for only 70% of the dataset, the decision rule was verified on the other 30% (test group).

The prediction results for both training and testing sets are shown below:

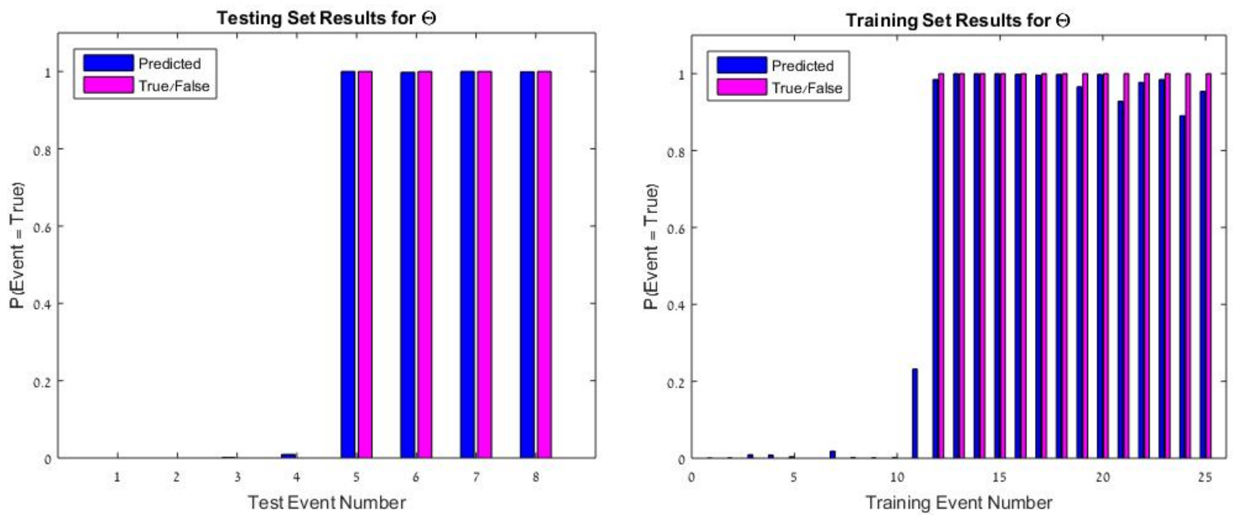


Figure 12 – Algorithm prediction results

One can see that the classification of the testing set is very good.
As expected (shown for visualization purpose only), the predicted values of the training set are equal to the real event type – because $\underline{\theta}$ was trained for this set specifically.

6 Implementation

6.1 Software Description

This chapter describes the software modules used in the glass breakage detector based on STM32L476RG with CCA02M1 - MEMS microphones module.
Figure 13 describes the high-level program flow of the application. After the initialization process the device enters to "waiting for event" mode. In this mode, every new sampled data of 100ms interval is filtered and processed to detect thud sound[6]. If thud occurs, the device enters to "glass breakage detection" mode during which the audio from the microphone is been processed, and using logistic regression the device decides if breakage occurred. In case of glass breakage detection comes back positive, alarm is turned on, otherwise the device simply returns to "wait for event" mode.

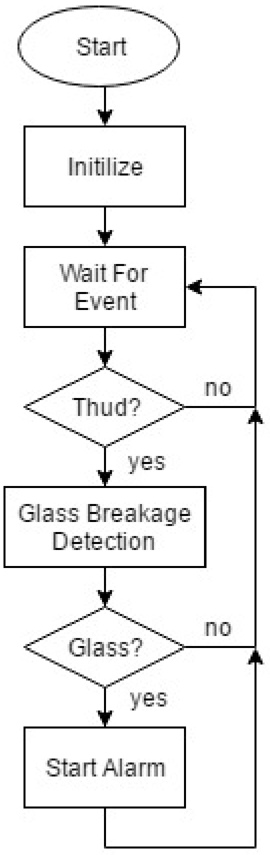


Figure 13 – Firmware base flow diagram

Initialization

This section of the code runs from the main function of the program and initializes the system and all the peripherals needed before moving to the next phase- *wait for event.*
The first step in this section is configuring the system PLL clock. This is a clock which fed by "Multispeed Internal RC Oscillator" (MSI). the RC runs at 40MHz and the system clock runs at 80MHz (x2 multiplication of RC). The clocks prescaler for all CPU busses (AHB, APB) is 1.
During this section, the IIR lattice filters which needed in the *wait for event* and the *glass breakage detection* modes, are initialized with the right coefficients to create 35Hz, 300Hz, 5kHz and 10kHz bandpass filters (coefficients can be found in the end of this section). Their state buffers also initialized to zero.
The board's built-in LED is initialized to use GPIO pin number 5 in *Output Push Pull* mode.
Finally, the DFSDM channel (number 2) and filter (number 0) are initialized: For the channel, GPIO pin 2 is used for the microphone's clock and GPIO pins 10,14 for its PDM ports. All the GPIOs are set to *Alternate Function Push Pull* mode. For the filter, DMA1 channel 4 is selected.
In order to move to *Wait for event* mode, *eventFlag* is set to 0 and DFSDM conversion is initialized.

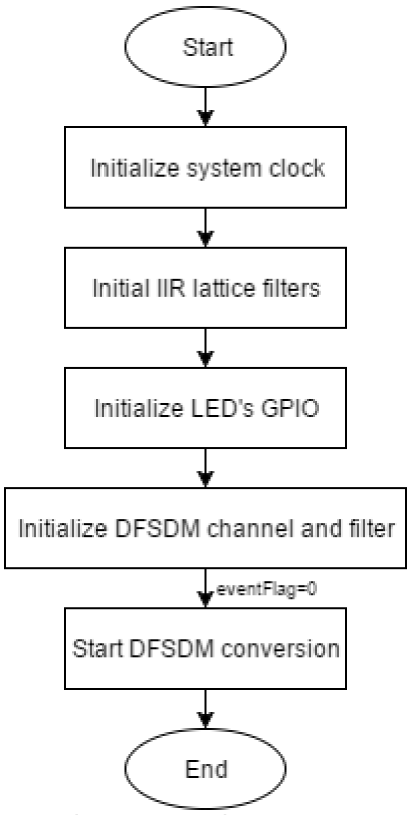


Figure 14 – Init sequence

Wait for Event – Thud Detection

After initiating the DFSDM conversion while *eventFlag* is set to 0, the device works on *Wait for event* mode. In this mode, the DFSDM conversion uses the *WaitBuff* as a recording buffer which is 3200 samples long, this means that every 100ms new raw data is received from the DFSDM. The raw data first passes through HP filter to eliminate any DC signal. The data then transferred through IIR lattice bandpass filter with $f_0 = 300Hz$. This filter is used to recognize the thud sound which usually has this low-frequency component. From the corresponding output samples of the filter RMS value is calculated. The glass breakage detection algorithm is initialized only if this RMS value exceeds a prefixed threshold. In this case, the DFSDM conversion, which based on *WaitBuff,* stops, the *eventFlag* is set to 1 and DFSDM conversion, which uses *AnalyzeBuff* as the recording buffer, is re-initialized to enter the *Glass breakage detection* mode.

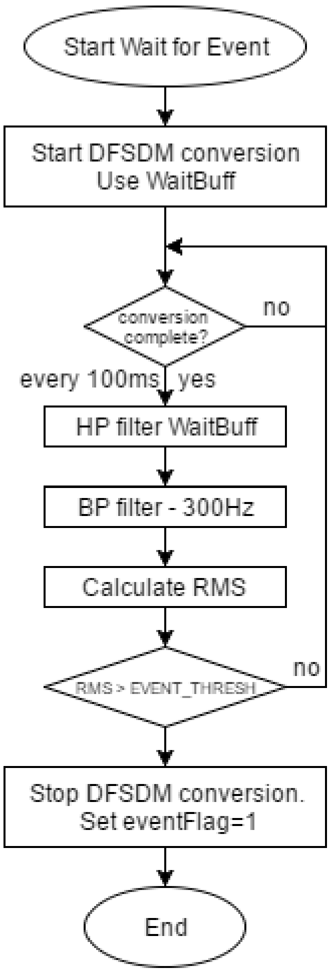


Figure 15 – Thud detection flow diagram

Glass Breakage Detection

In case of valid thud sound detection, the system enters to *Glass breakage detection* mode. To initiate this mode, *eventFlag* is set to 1 and DFSDM conversion is started using *AnalyzeBuff* as a recording buffer. The *AnalyzeBuff* is 32 samples long, thus every 1ms interval new raw data, which was recorded from the microphone, is ready to be processed. Each block of 32 samples is filtered using four different IIR lattice bandpass filters: 35Hz, 300Hz, 5kHz and 10kHz. The four different outputs of the filtered 1ms long interval are saved into the *FeaturesBuff* array which can hold up to 100ms of filtered data (3200 samples). The time that takes the device to filter the data is shorter than 1ms thus, by using DMA, the recording and processing can occur simultaneously. When *FeaturesBuff* fills up, RMS values are calculated to create four new features, one from each filter, which will be used in the logistic prediction function. Otherwise, if *FeaturesBuff* is not full, the program simply returns to record and process a new block from the DFSDM. This cyclic process takes place over a total time of 2 seconds to produce 80 features, 20 from each one of the four filters. Those features are saved in the *IIR_feautres* array which is used by the logistic prediction function to decide whether glass breakage occurred or not. In case of glass breakage detection comes back positive, alarm is turned on, otherwise the device simply returns to "wait for event" mode.

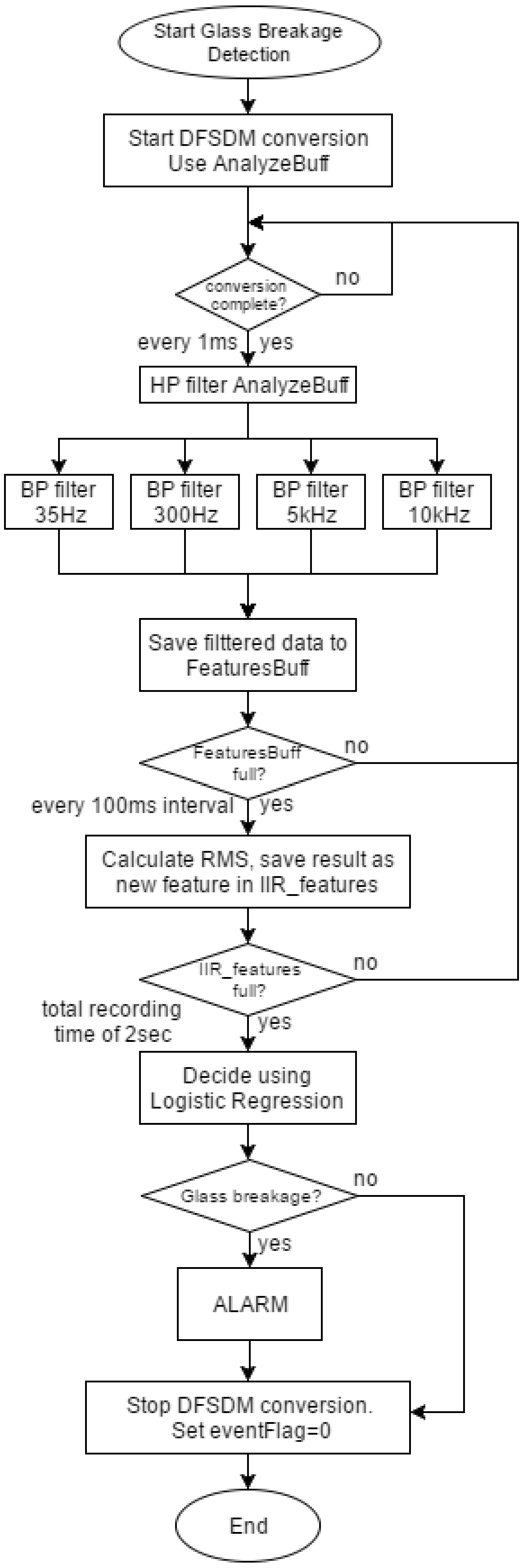
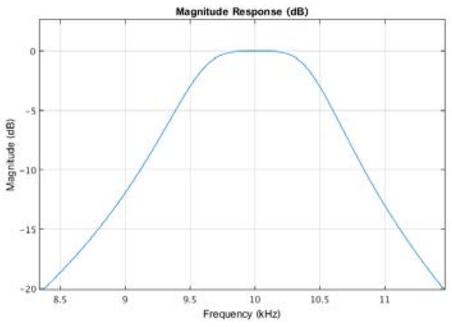

Start Glass Breakage Detection
Start DFSDM conversion Use AnalyzeBuff
conversion complete?
no
every 1ms
yes
HP filter AnalyzeBuff
BP filter 35Hz
BP filter 300Hz
BP filter 5kHz
BP filter 10kHz
Save filttered data to FeaturesBuff
FeaturesBuff full?
no
every 100ms interval
yes
Calculate RMS, save result as new feature in IIR_features
IIR_features full?
no
total recording time of 2sec
yes
Decide using Logistic Regression
Glass breakage?
no
yes
ALARM
Stop DFSDM conversion. Set eventFlag=0
End

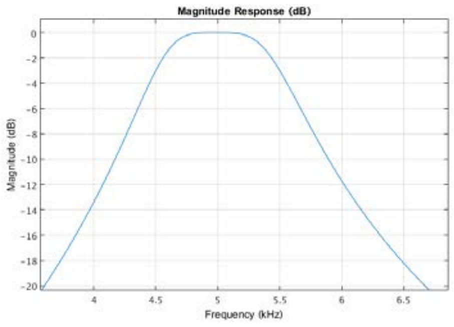
Figure 15 – Glass breakage detection flow diagram

IIR lattice filters details

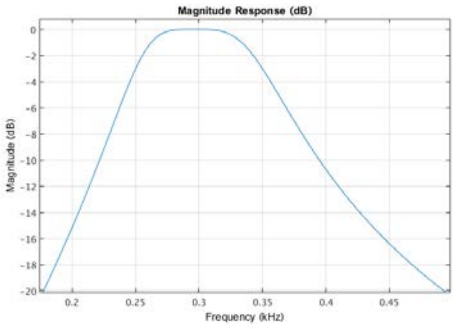
10kHz bandpass filter lattice coefficients:
K: 24823, 12358, 32143, 12841
V: 277,-396, -719, 718,640



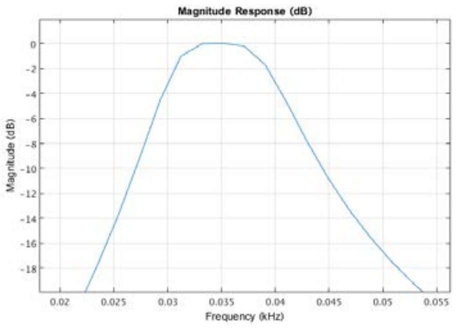
5kHz bandpass filter lattice coefficients:
K: 24823, -17942, 32149, -18638
V: 277,575, -373, -840,271



300Hz bandpass filter lattice coefficients (used also in thud detection):
K: 31871, -32706, 32762, -32718
V: 3, 12, 12



35Hz bandpass filter lattice coefficients:
K: 32723, -32767, 32767, -32767
V: 1



7 Training Sequence

7.1 Setup

The first step in the training procedure was to build a database of glass breakage sounds and other daily life sound events. These type of sound recordings available from various sources such as glass breakage detectors application notes, free sounds online databases and so on.

For practical reasons, the setup that was used in the training sequence was the NUCLEO board with the CCA02M1 extension board (using single microphone).

The next step was to play these sounds to the detector and record the algorithm output, before the event estimation (e.g. the sound features). Each feature vector was saved with its type – 0 for false event and 1 for true.

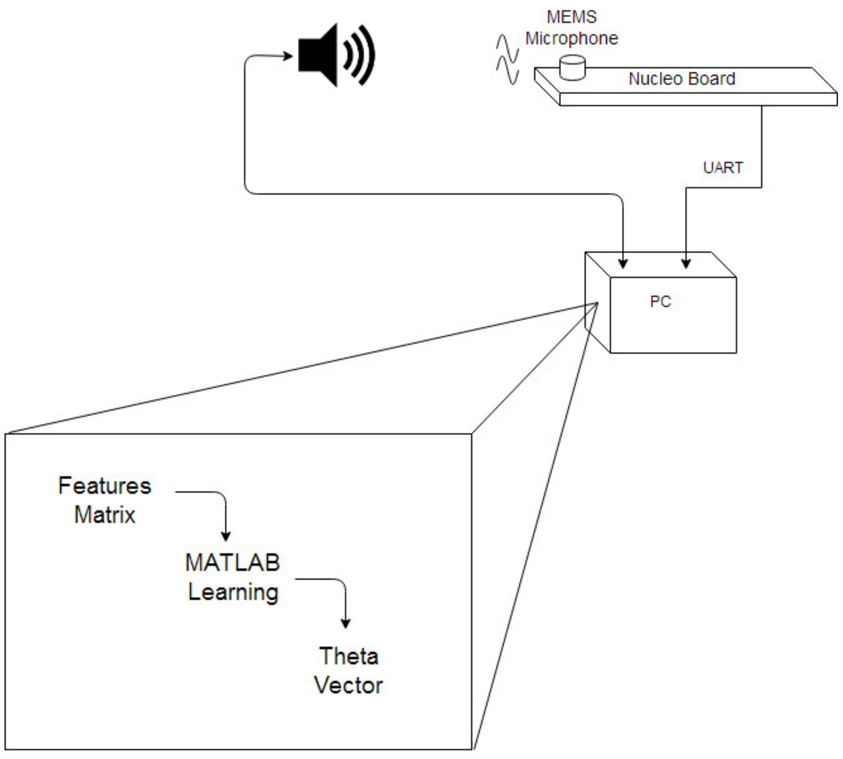


Figure 16 - Learning sequence illustration

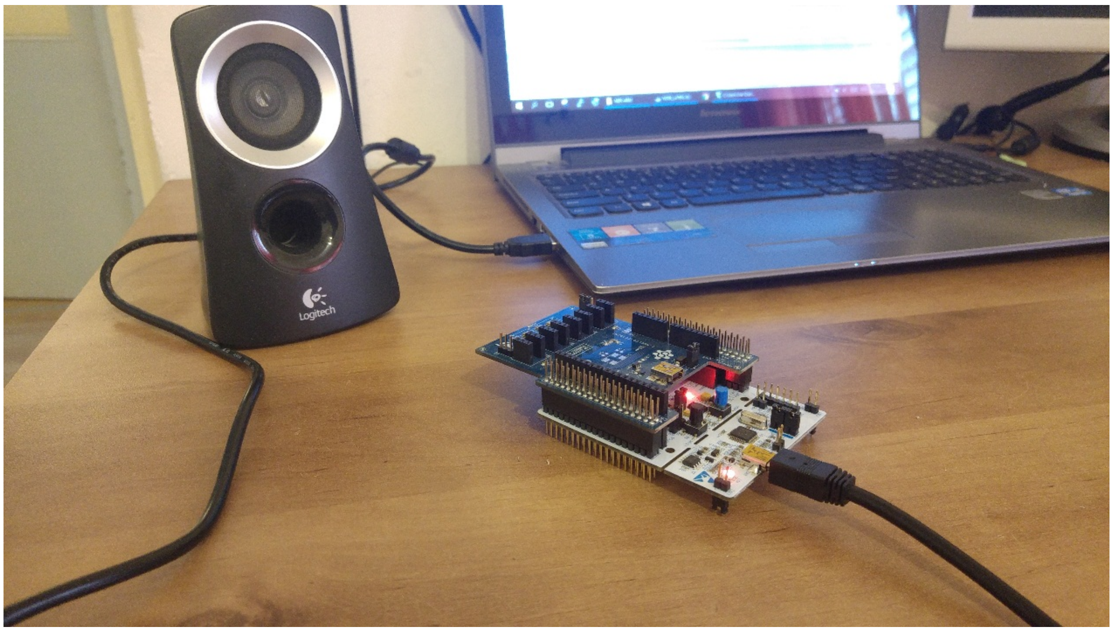
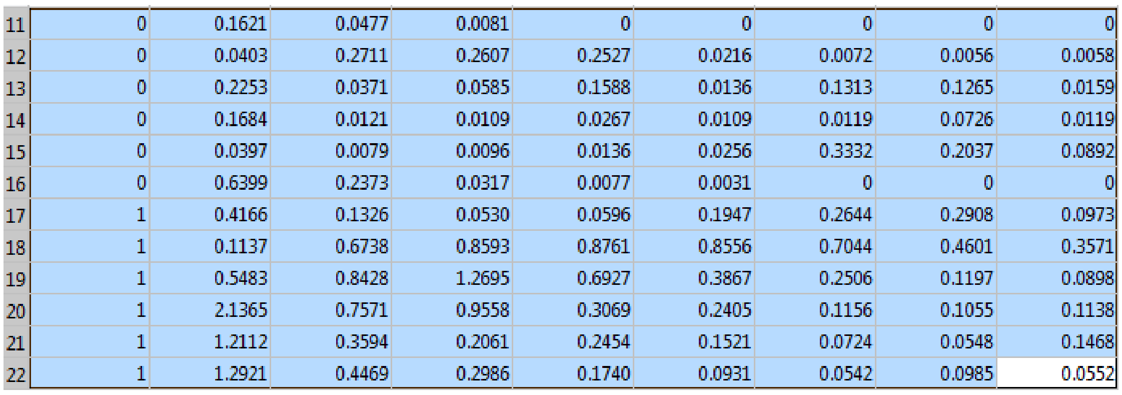


Figure 17 - Learning sequence photo

After the features matrix was built, it was used as input to the logistic regression fitting algorithm.
The following figure shows part of the features matrix:



11	0	0.1621	0.0477	0.0081	0	0	0	0	0
12	0	0.0403	0.2711	0.2607	0.2527	0.0216	0.0072	0.0056	0.0058
13	0	0.2253	0.0371	0.0585	0.1588	0.0136	0.1313	0.1265	0.0159
14	0	0.1684	0.0121	0.0109	0.0267	0.0109	0.0119	0.0726	0.0119
15	0	0.0397	0.0079	0.0096	0.0136	0.0256	0.3332	0.2037	0.0892
16	0	0.6399	0.2373	0.0317	0.0077	0.0031	0	0	0
17	1	0.4166	0.1326	0.0530	0.0596	0.1947	0.2644	0.2908	0.0973
18	1	0.1137	0.6738	0.8593	0.8761	0.8556	0.7044	0.4601	0.3571
19	1	0.5483	0.8428	1.2695	0.6927	0.3867	0.2506	0.1197	0.0898
20	1	2.1365	0.7571	0.9558	0.3069	0.2405	0.1156	0.1055	0.1138
21	1	1.2112	0.3594	0.2061	0.2454	0.1521	0.0724	0.0548	0.1468
22	1	1.2921	0.4469	0.2986	0.1740	0.0931	0.0542	0.0985	0.0552

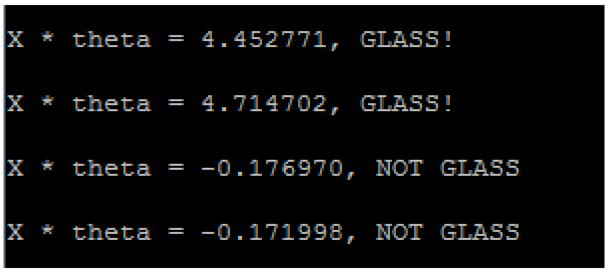
Figure 18 - Part of the features matrix

The full size of the features matrix is 34 rows X 81 columns, which means 34 sound events, each consist of 80 features (20 per filter) plus the event type.
The algorithm output is the vector θ which is hard coded in the glass breakage detector flash.

This vector will be used from now on to classification of sound events.

7.2 Results

The theoretical results for the sound classification as calculated by MATLAB are shown in the algorithm document.
In order to test the performance in real life, the same samples was now played to the detector only this time the estimation algorithm is running with the estimation step and its output is the value of the multiplication between theta and the features vector: $X \cdot \theta^T$.



```
X * theta = 4.452771, GLASS!

X * theta = 4.714702, GLASS!

X * theta = -0.176970, NOT GLASS

X * theta = -0.171998, NOT GLASS
```

Figure 19 - Algorithm classification

It is important to notice that many false events do not trigger the process due to the lack of energy in the thud frequencies (250-350 Hz), but for presentation purposes we've made them to trigger the algorithm.
A comparison of real life performance is shown below:

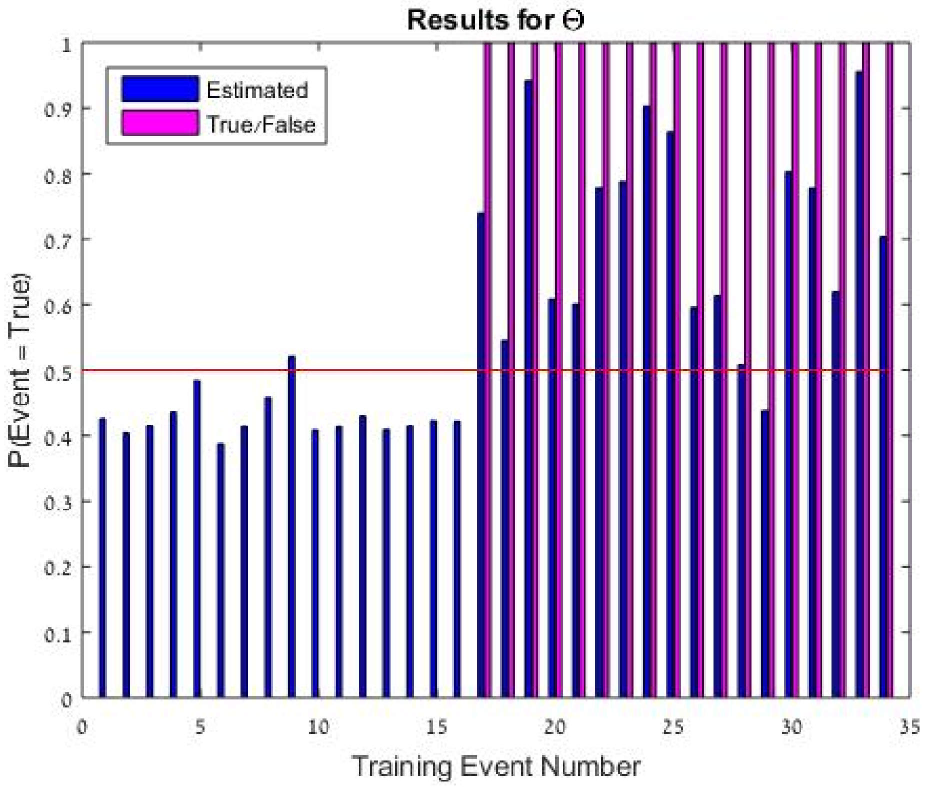
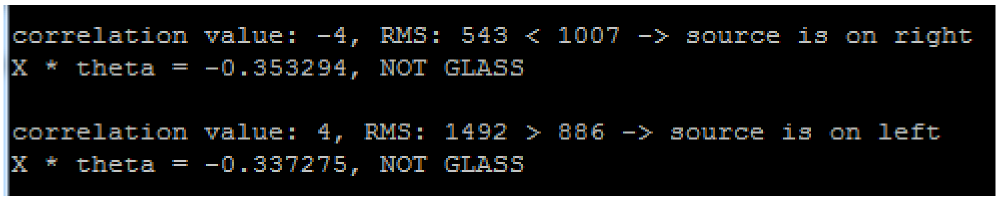


Figure 20 – Real life performance

We can see from the plot that if we set the threshold to 0.5 then we get false alarm for one sample and miss detect for one sample, which is a good compromise.
This behavior is probably due to the lower audio quality of these particular samples.
In real life applications, only verified high quality audio sources will be used and thus the performance would be much better.

7.3 Using Sound Localization

When using two microphones and when the correlation feature is enables, the output for false event in two different directions is shown:



```
correlation value: -4, RMS: 543 < 1007 -> source is on right
X * theta = -0.353294, NOT GLASS

correlation value: 4, RMS: 1492 > 886 -> source is on left
X * theta = -0.337275, NOT GLASS
```

Figure 21 - Source direction estimation

It is possible to use the sound localization method to improve the detector performance, although it is not supported by the Simba board.

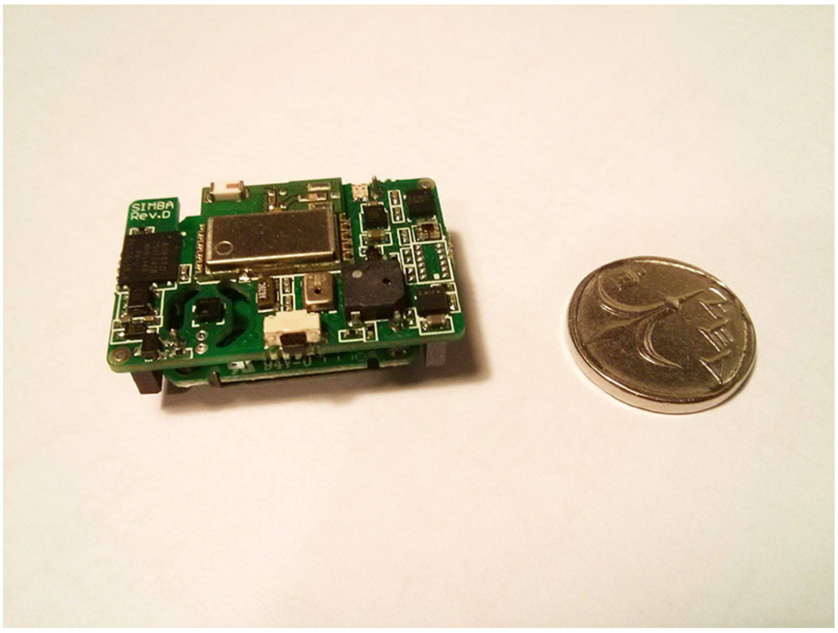


Figure 22 – Simba board uses single microphone

8 Power Optimization

8.1 Introduction

In almost every application that runs on batteries, power saving is crucial on the users side. Power consumption performance can make the difference between a good and excellent device.
Since the glass breakage detector run on batteries, it is necessary to optimize its power consumption by choosing the best hardware configuration and firmware flow/run time.

The scope of this document is to describe the power consumption and power configurations of the glass breakage detector system.

8.2 Power Configuration

The core of the glass breakage detector is an ARM Cortex M4 microcontroller – ST32L476, which is intended for ultra-low power applications by its manufacture.

Supply Voltage – For the Simba board, entire circuitry designed to be powered from 3V coin battery. NUCLEO board is powered by USB and used for evaluation only.

Main Clock - The ST32L476 clock source is internal RC oscillator which runs at 48 MHz. Such high speed clock is necessary for filtering and floating point math done in real time. During the algorithms analysis, the calculations takes 50% of the run time if the audio packets are coming every 1ms.
Since the analysis step happens only after thud detection, it can be neglected for power consumption calculations (e.g. the relative amount of time in this state is very small).

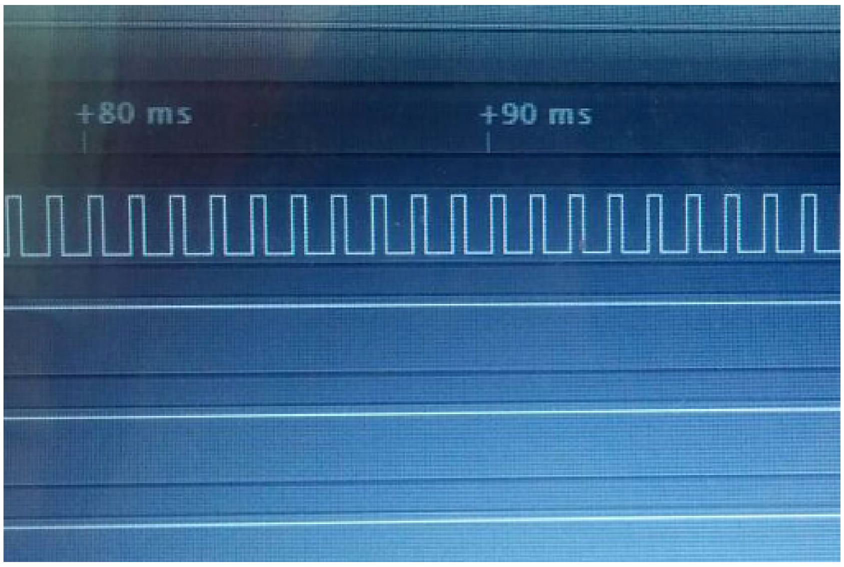


Figure 23 - Run time during analysis state – 50%

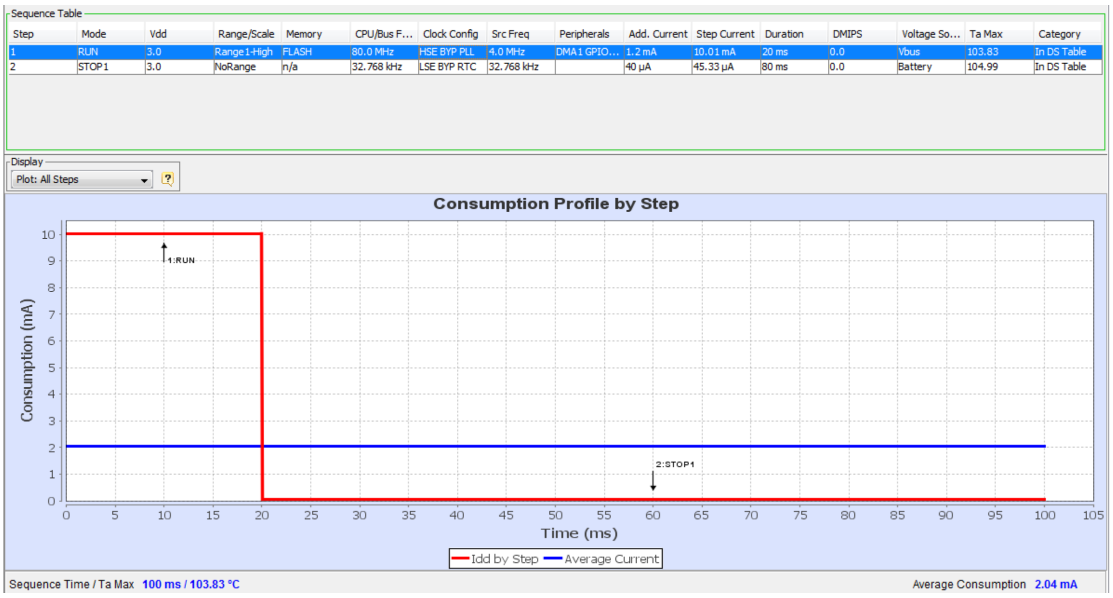
Low Frequency Clock - For power saving purpose, we use the low frequency 32 kHz clock (LSI RC). This clock is used during the detection state – the MCU is in deep sleep mode and wakes up every 100ms to sample the environmental sounds for thud detection.

Steady State Power Consumption Calculation
MCU steady state operation:

- Running from low frequency clock during sleep.
- Running from 48 MHz clock during run time.
- Wake up every 100ms for thud detection.
- Detection time is 20ms.

The typical current consumption in runtime is 118 uA/MHz[7] (for Coremark code) which is approximately 9.4 mA for the MCU only. For this value we need to add the typical current consumption of the MP34DT01[8] MEMS microphones in active mode – 1.2 mA (0.6 mA each). In total we've got 10.6mA in active mode.
The current values for stop state are 5uA for the MCU and 40uA for the two microphones. In total – 45uA.

The next figure shows the expected average power consumption calculated by STM32CubeMX calculator:



Sequence Table

Step	Mode	Vdd	Range/Scale	Memory	CPU/Bus F...	Clock Config	Src Freq	Peripherals	Add. Current	Step Current	Duration	DMIPS	Voltage So...	Ta Max	Category
1	RUN	3.0	Range1-High	FLASH	80.0 MHz	HSE BYP PLL	4.0 MHz	DMA1 GPIO...	1.2 mA	10.01 mA	20 ms	0.0	Vbus	103.83	In DS Table
2	STOP1	3.0	NoRange	n/a	32.768 kHz	LSE BYP RTC	32.768 kHz		40 µA	45.33 µA	80 ms	0.0	Battery	104.99	In DS Table

Display
Plot: All Steps

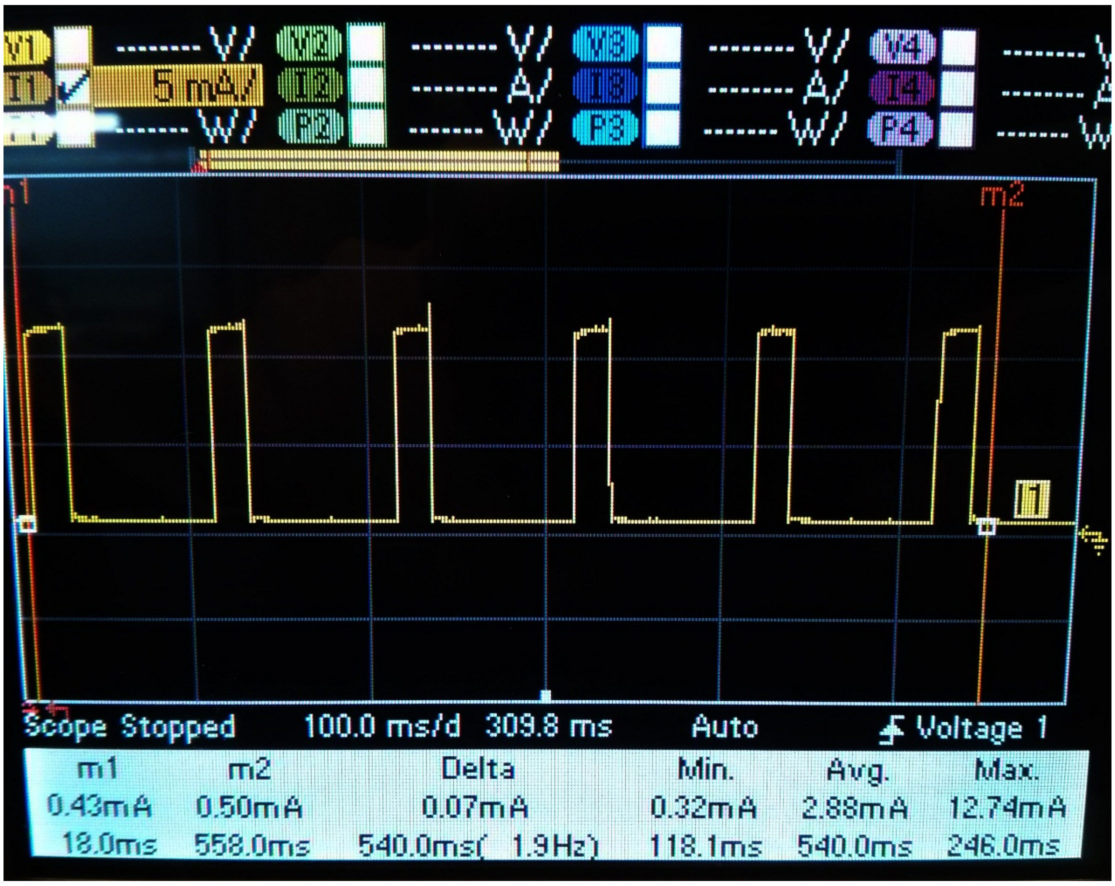
Sequence Time / Ta Max 100 ms / 103.83 °C
Average Consumption 2.04 mA

Figure 24 - STM32CubeMX power consumption calculator

We can see that due to the long sleep period the average current is 2 mA.

8.3 Measurements

A current measurement of the MCU in steady state were made using power analyzer. The measurement were made on the NUCLEO development board with the CCA02M1 extension board, as instructed in the user manual.



m1	m2	Delta	Min.	Avg.	Max.
0.43mA	0.50mA	0.07mA	0.32mA	2.88mA	12.74mA
18.0ms	558.0ms	540.0ms(1.9Hz)	118.1ms	540.0ms	246.0ms

Figure 25 - MCU current plot in 500 ms period

We can see from that the current consumption is very similar to the one simulated by STM32CubeMX, although the minimum current is 320uA – way above the expected value of 40uA. As a result, the average current is also higher – 2.88mA compared to 2mA in the simulation.
A possible explanation for this difference is that the debugger circuit or one of the LDO's are not entirely disconnected and thus consume some current.

8.4 Optimization Options

An average current consumption of 2 mA indicates poor performance of any device running on batteries. Since the capacity of a coin battery is approximately 250mAh, it will only last for 125 hours and possibly less.
In this section we will try to estimate the current consumption when using dedicated parts for ultra-low power application.

The first change we can make to our system is replacing the digital microphone with an analog one. Then we can use the DFSDM analog watchdog peripheral which can be programed to wake up the MCU when a predefined threshold of the microphone energy is passed. A possible microphone model is the MP23AB02B[9].

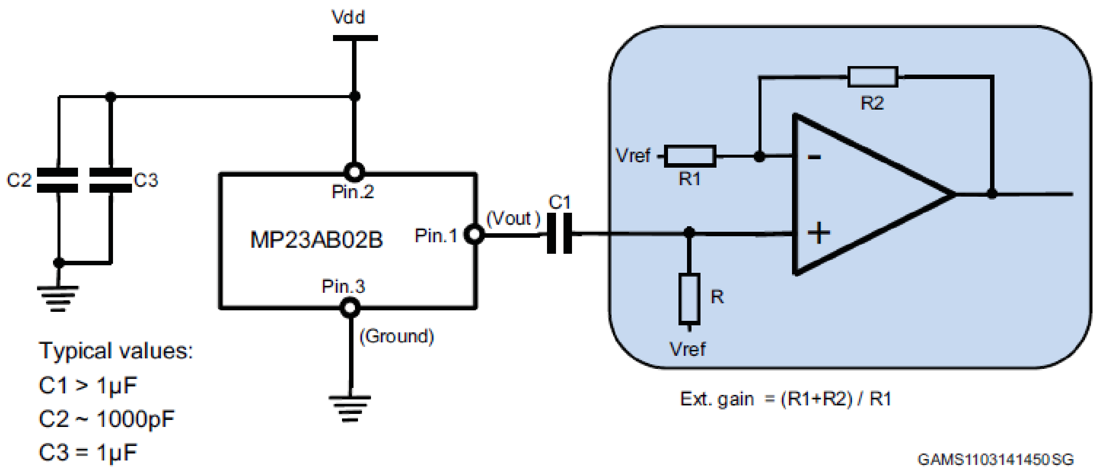
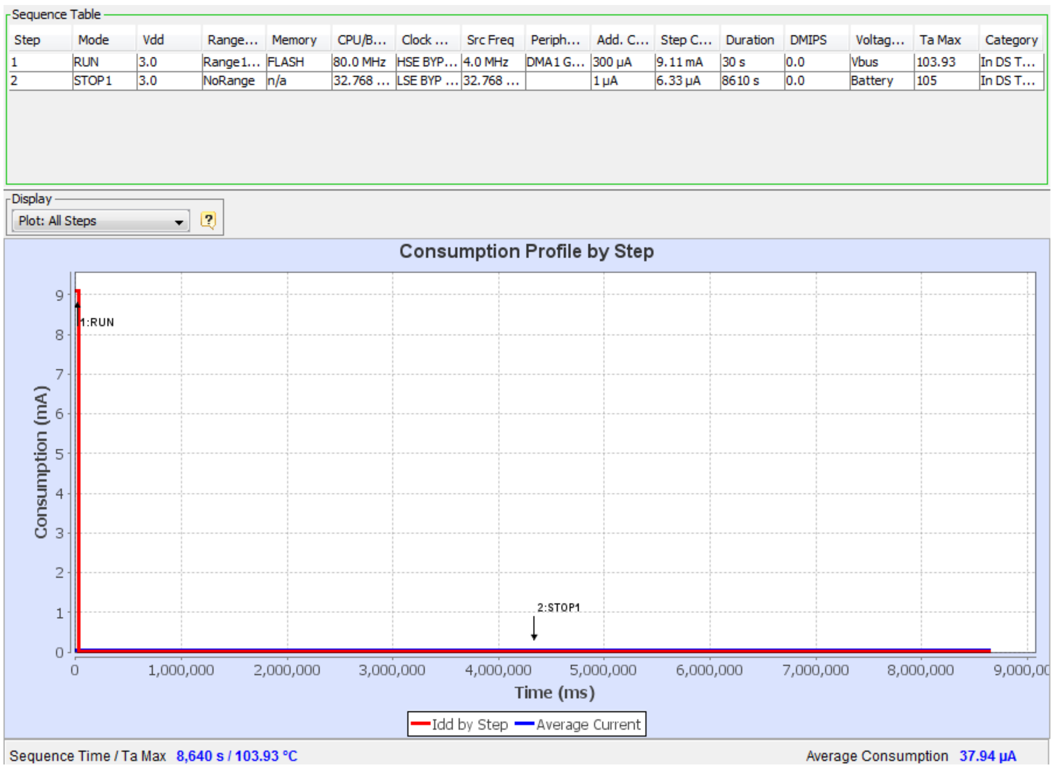


Figure 26 – Example MP23AB02B circuit

In this case, the MCU only wakes up when an unusual sound event occurs and not every constant amount of time like before. The tradeoff of this configuration is the additional amplifier.
If the analog watchdog threshold is calibrated on the fly it is possible to estimate during run time if a situation in which the MCU wakes up repeatedly is due to normal activity or not. These methods will allow as minimum as possible false wakeups.

For modeling purpose we will assume that the number of false wakeups rate is 300 wakeups per 24 hours. If for every wakeup the MCU is running 100ms to detect thud and to update the watchdog threshold, we get in total 30 seconds a day in active mode.



Sequence Table

Step	Mode	Vdd	Range…	Memory	CPU/B…	Clock …	Src Freq	Periph…	Add. C…	Step C…	Duration	DMIPS	Voltag…	Ta Max	Category
1	RUN	3.0	Range1…	FLASH	80.0 MHz	HSE BYP…	4.0 MHz	DMA1 G…	300 µA	9.11 mA	30 s	0.0	Vbus	103.93	In DS T…
2	STOP1	3.0	NoRange	n/a	32.768 …	LSE BYP …	32.768 …		1 µA	6.33 µA	8610 s	0.0	Battery	105	In DS T…

Display: Plot: All Steps

Sequence Time / Ta Max 8,640 s / 103.93 °C

Average Consumption 37.94 µA

Figure 27 - STM32CubeMX power consumption calculator

We can see that the average current consumption dropped to 38uA. For this current we need to add the steady state current consumption of the additional amplifier.
A possible amplifier is from the ST94x product line, which have 1.2uA supply current [10] with no load.
If we consider the 30 seconds of active state during 24 hours period (500uA current), we have an average current of 1.4uA.
Thus, for this configuration and timing, the additional amplifier is negligible in power considerations.

In total, the average current consumption is 40uA which allows the device to run on a coin battery for approximately 8 months.

9 Results Analysis

9.1 Compering Simulation and Real life Performance

The following table summarize the tests results (real time performance):

Test	**Simulation**	**Real Life Performance**
Glass Breakage Detection *Success in event classification*	100%	94%
Sound Localization *distinguish sound source between two directions*	100%	70%
Power Consumption	2 mA	2.9 mA

The tests results are very good for a prototype. It is important to remember that the tests were not made in optimal conditions or dedicated circuitry. Thus we are sure that for dedicated product, the performance will be much better.

10 Conclusion

By examining the final test results of the projects different aspects we can see that the goals achieved:

- Event Classification – even though the accuracy is not 100%, it is not very far from it. Since the project goal was to prove that our solution is acceptable and not to provide a high-end product, working out of the box – we can definitely say that this part performance are very good.
- Sound Localization – This is not the primary module of the event classification process but an additional one that can only improve the accuracy of the algorithm. While it is not flawless, its performance are very good considering that only two microphones are used, in a relatively low sampling frequency.
- Power Consumption – The systems full potential can be seen from the projects measurements and simulation. The maximal performance could only be achieved using dedicated circuit design and by using evaluation boards.

In addition, there a several ways to improve the system performance:

- Using more complex algorithms for detecting glass breakage pattern. Some of the methods introduced in the theoretical background chapter. The main tradeoffs of using these methods are the computation time and power consumption.
- Adding external circuitry for thud detection. This way the cost of searching the thud and analyzing it will not affect the MCU. The tradeoff of this external layer is cost due the additional components. Another issue that might appear is the steady state power consumption of such circuit. Thus, the circuit design must minimized the power consumption so it will be practical.
- Adding more microphones to the sound localization step will increase the resolution of the sound source direction. In this way, the false alarm probability will be minimized. The tradeoffs of using more microphones are the cost and the delay the additional delay due to the longer calculations (cross correlation between every pair of microphones).
- In order to make the application more practical, it is possible to add an additional alarm system layer that could inform the user even if he’s not in his home. This layer could be implemented by connection the MCU via Bluetooth to another unit that is connected to the internet. Bluetooth connectivity is supported by the Simba board and thus will not require additional components.

11 Bibliography

Application Notes:

[1] Bhargavi Nisarga, Kripasagar Venkat, "A Robust Glass-Breakage Detector Using the MSP430", TI Application Report SLAA389, February 2008

http://www.ti.com/lit/an/slaa389/slaa389.pdf

[2] Vadym Grygorenko, "Consumer of Industrial: Acoustic Glass Break Detector", Cypress Application Note AN2186
http://www.cypress.com/file/141276/download

[3] “Tutorial for MEMS Microphones”, ST, AN4426
http://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

Articles:

[4] Kenji Suyama, Kota Takahashi, "A Talker Tracking Method Using Two Microphones Based On the Sound Source Localization", Faculty of Engineering, Tokyo DENKI University.

http://kilyos.ee.bilkent.edu.tr/~signal/defevent/papers/cr1630.pdf

[5] "Demystifying Delta-Sigma ADCs", Maxim Integrated, Tutorial 1870
http://pdfserv.maximintegrated.com/en/an/AN1870.pdf

Books

[6] Donald S. Reay. “Digital Signal Processing Using the ARM Cortex-M4”, Wiley, 2016

Datasheets:

[7] “STM32L4xx advanced ARM-based 32-bit MCUs”, ST Reference Manual RM0351

http://www.st.com/content/ccc/resource/technical/document/reference_manual/02/35/09/0c/4f/f7/40/03/DM00083560.pdf/files/DM00083560.pdf/jcr:content/translations/en.DM00083560.pdf

[8] MP34DT01-M, “MEMS audio sensor omnidirectional digital microphone”, ST

http://www.st.com/content/ccc/resource/technical/document/datasheet/47/bd/d2/13/8d/fd/48/26/DM00121815.pdf/files/DM00121815.pdf/jcr:content/translations/en.DM00121815.pdf

[9] MP23AB02B, “MEMS audio sensor high-performance analog bottom-port microphone”, ST

http://www.st.com/content/ccc/resource/technical/document/datasheet/08/26/0f/ff/5c/36/44/36/DM00111230.pdf/files/DM00111230.pdf/jcr:content/translations/en.DM00111230.pdf

[10] TS941, “Ultra-micro power amplifier with CMOS inputs”, ST

http://www.st.com/content/ccc/resource/technical/document/datasheet/23/d5/53/9b/08/82/4c/a6/CD00002000.pdf/files/CD00002000.pdf/jcr:content/translations/en.CD00002000.pdf