TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

FACOLTÀ DI INGEGNERIA

Laurea Magistrale in Ingegneria Elettronica

CAR-AUDIRE: an IoT Based System for Collaborative Real-Time Noise Mapping

Relatore:
Prof. Marco Re

Candidato:
Sergio Canalicchio

Correlatori:
Prof. Alberto Nannarelli
Dott. Vittorio Colombo

Anno Accademico 2018/2019

$\backslash$m/

Contents

Chapter 1

Introduction

Nowadays, noise exposure monitoring and its reduction are among the main concerns for citizens, politicians, administrations, and technical-scientific bodies because a long-ter, exposure to excessive noise levels can affect human health.
Directive 2002/49/EC “relating to the assessment and management of environmental noise” (Environmental Noise Directive, END) is the last effort to harmonise European member states’ policies and technical approaches concerning noise exposure reduction issues. Noise mapping of relevant environmental noise sources in agglomerations, fol- lowing the directive, has presented the opportunity for assessing noise levels in main European agglomerations.
Noise mapping is a very complex and challenging issue: noise sources are widely diffuse, especially in urban areas; their characterisation is not sim- ple and the quantification of a citizen’s exposure to noise is a very difficult task. Moreover, the process requires different skills and experts: from geo- spatial analysis to uncertainty evaluation and psychoacoustic approaches to maximise cost benefits and action plan adherence.

The directive recommends that wherever possible noise mapping should be carried out by computation (simulation method). One limitation of this method is that it can not inform about incidental or short-term variations in environmental noise exposure. For these reason, research project focused on development of systems suitable for real-time noise maps production. This is achieved by implementing a noise monitor network which continuously collect noise data and transmits them to a data centre in which a noise mapping software is running. many research project focused on using smartphone as nodes of the noise monitor network implementing a participatory sensing enabling citizen to measure their personal exposure to noise. The main limitation of this approach is the measurement accuracy due to the infeasibility to use the same calibration algorithm for all the smartphone.

CAR-AUDIRE project aims to using cars as distributed sensors able to make acoustic measurement that, sent by IoT wireless communication standards, are used to create real-time noise maps in cities as well as to recognize and react to sound event.

This thesis aim to implement an environmental noise measurement system in the framework of CAR-AUDIRE project.

In Chapter 2 we give an overview about environmental noise and the directive that manage the exposure levels to environmental noise. Moreover, it shows the methodologies to carry out noise maps describing some research project about it.

Chapter 3 describes the CAR-AUDIRE project.

Chapter 4 describes the Sound Level Meter (SLM) algorithm, showing all the functional block and paying attention at the kind of measure and standard specification.

Chapter 5 describe how to implement the weighting filters given in the standard.

In Chapter 6 we show the system implementation. In particular we describe the SODAQ SARA AFF R410M that allow a fast prototyping approach to test the noise measurement and data transmission.

Chapter 7 shows the power consumption measurement set up.

Chapter 2

Environmental Noise

Noise is usually defined as unwanted or undesired sound. Although people tend to get used to noise exposure, the level of habituation is never complete and differs for individuals. Long-term exposure to excessive noise level can affect human health. Negative health effect were first identified in people exposed to excessive occupational noise levels, i.e. the noise people are exposed to within their job. Excessive exposure to occupational noise may occur in many industries, for example manufacturing, construction and transport. High exposure to noise levels in the workplace was related with noise-induced hearing damage and even hearing loss. For this reason, authorities have decreed regulations to asses and limit occupational noise.

Over the last two decades research focus has extend to environmental noise which is defined as [1, *Article 3(a)*]:

> «unwanted or harmful outdoor sound created by human activities, including noise emitted by means of transport,

road traffic, rail traffic, air traffic, and from sites of industrial activity»

Environmental noise has always been an important problem for people. Already in ancient Rome, chariots were banned from pavement because the noise of their wheels was causing disruption of sleep and annoyance to the Romans. For the same reason, in some cities in Medieval Europe horse carriages were not allowed during night time. However, the noise problems of the past are not comparable with those of modern society where the increasing of environmental noise is associated whit technological progress, the development of transport and industry and the urbanization process. People are exposed to this kind of noise in their daily lives: in urban areas, major environmental noise sources are road traffic, rail traffic, air traffic and industrial activity.
Over the recent decades, studies on environmental noise show that a long-term exposure to this kind of noise affect human behaviour, well-being, productivity and health.
The *World Health Organization* (WHO) played an important role in regulating environmental noise exposure. In [2], the WHO sets the exposure thresholds to not be exceeded as precaution against health problems and to avoid disturbance of normal activities in local communities: during the day and the evening the noise levels should stay below 55 $dB(A)$ outdoors and 50 $dB(A)$ inside dwelling. To avoid distruption of sleep the WHO established that during the nigh the noise levels should stay below 45 $dB(A)$ outdoors and below 30 $dB(A)$ inside bedrooms. In [3] the WHO lowered the outside night threshold to 40 $dB(A)$, but advised policymakers to for 50 $dB(A)$ in situations where $40dB(A)$ is not feasible.
The negative health outcomes due to long-term exposure to environmental noise concern both auditory and non-auditory effects. The range of non-auditory effects includes annoyance, sleep disturbance, cardiovascular disease, and impairment of cognitive performance in children. In [4], WHO summarize the evidence on the relationship between environmental noise and these health effects.

2.1 Directive 2002/49/EC: Environmental Noise Directive

Environmental noise has always been the source of many complaints from the public but action to reduce it has had a lower priority than that taken to address other environmental problems such as air and water pollution.
The Green Paper on Future Noise Policy was the first step of the European Commission to develop a noise regulation program. It aimed to stimulate public discussion on the approach to noise policy [5]. The next big step was taken in 2002 when the European Parliament and Council implemented the *Directive 2002/49/EC* relating to the assessment and management of environmental noise, also known as *Environmental Noise Directive* (END). The aim of the END is [1, *Article 1*]:

> «to define a common approach intended to avoid, prevent or reduce on a prioritised basis the harmful effects, including annoyance, due to exposure to environmental noise»

For this purpose, the END sets out the following actions to be implemented progressively:

- the determination of exposure to environmental noise, through noise mapping, by methods of assessment common to the Member States
- ensuring that information on environmental noise and its effects is made available to the public
- adoption of action plans by the Member States, based upon noise-mapping results, with a view to preventing and reducing environmental noise where necessary and particularly where

exposure levels can induce harmful effects on human health and to preserving environmental noise quality where it is good

Member States must ensure the data to be sent to the Commission are collected in accordance with *Articles 7, 8* and *Annex VI* of the END as follows:

- for agglomerations (part of a territory whit a population over 100 000):
  - the estimated number of people (in hundreds) living in dwellings that are exposed to each of the following bands of values of L_{den} in dB 4 m above the ground on the most exposed facade: $55-59$, $60-64$, $65-69$, $70-74$, >75, separately for noise from road, rail and air traffic, and from industrial sources [1, *Annex VI(1.5)*]
  - 'the estimated total number of people (in hundreds) living in dwellings that are exposed to each of the following bands of values of L_{night} in dB 4 m above the ground on the most exposed faccade: $50-54$, $55-59$, $60-64$, $65-69$, >70, separately for road, rail and air traffic and for industrial sources' [1, *Annex VI(1.6)*]
  - the strategic noise maps that graphically represent this data; these maps must show at least the 60, 65, 70 and 75 dB contours
  - an action plan, designed to manage noise issues and effects, including noise reduction if necessary, and to protect quiet areas against an increase in noise
- for major roads (which have more than 3 000 000 vehicle passages a year), major railways (which have more than 30 000 train passages per year) and major airports (civil airports which have more than 50 000 movements[1] per year):

[1]A movement being a take-off or a landing.

- the estimated total number of people (in hundreds) living outside agglomerations in dwellings that are exposed to each of the following bands of values of L_{den} in dB 4 m above the ground and on the most exposed facade: $55-59$, $60-64$, $65-69$, $70-74$, > 75 [1, *Annex VI(2.5)*]
- the estimated total number of people (in hundreds) living outside agglomerations in dwellings that are exposed to each of the following bands of values of L_{night} in dB 4 m above the ground and on the most exposed facade: $50-54$, $55-59$, $60-64$, $65-69$, > 70 [1, *Annex VI(2.6)*]
- the total area (in km^2) exposed to values of L_{den} higher than 55, 65 and 75 dB respectively. The estimated total number of dwellings (in hundreds) and the estimated total number of people (in hundreds) living in each of these areas must also be given [1, *Annex VI(2.7)*].
  The 55 and 65 dB contours must also be shown on maps
- an action plan, designed to manage noise issues and effects, including noise reduction if necessary

All datasets, maps and action plans shall be updated at least every 5 years.

The preparation of strategic noise maps by the Member States is essential for the implementation of the Directive. These maps show the values of noise indicators suitable for assessment of exposure to environmental noise.
Within the END, the noise indicators to be applied are L_{den} and L_{night}. The day-evening-night level L_{den} in decibels (dB) is an annual noise indicator which describes the average day-evening- night-time A-weighted equivalent sound pressure level over a complete year, while the night-time noise indicator L_{night} in decibels (dB) describes the night-time A-weighted equivalent sound pressure level

over a complete year. L_{den} is given by the following equation

$$L_{den} = 10 \log \left[\frac{1}{24} \left(12 \cdot 10^{\frac{L_{day}}{10}} + 4 \cdot 10^{\frac{L_{evening}+5}{10}} + 8 \cdot 10^{\frac{L_{night}+10}{10}} \right) \right] \tag{2.1}$$

where:

- L_{day} is the A-weighted long-term average sound level as defined in ISO 1996-2: 1987, determined over all the day periods of a year; the day period is 12 hours
- $L_{evening}$ is the A-weighted long-term average sound level as defined in ISO 1996-2: 1987, determined over all the evening periods of a year; the evening period is 4 hours
- L_{night} is the A-weighted long-term average sound level as defined in ISO 1996-2: 1987, determined over all the night periods of a year; the night period is 8 hours

The default values for the day period are from 7.00 to 19.00 while evening and night default values are from 19.00 to 23.00 and 23.00 to 7.00, respectively.

The weighting factors 5 and 10 in Equation 2.1 are intended to account for the increase in annoyance at different periods throughout the day, hence the addition of 5 to the value of $L_{evening}$ and 10 to the value of L_{night} meaning that intensifications in evening-time and night-time noise are stricter with respect to limit values.

A crucial action for the management of environmental noise is ensuring that information on noise exposure and its effects is made available to the public. For this reason, the results of the process of noise mapping have been made available via a noise observation and information service for Europe (NOISE) which is maintained by the European Environment Agency (EEA)[2]. Example of the NOISE

[2]http://noise.eea.europa.eu/

service is shown in the figures below. Figure 2.1 shows L_{den} levels due to traffic noise in Milan while Figure 2.2 shows the night-time noise levels L_{night} due to traffic noise, too.

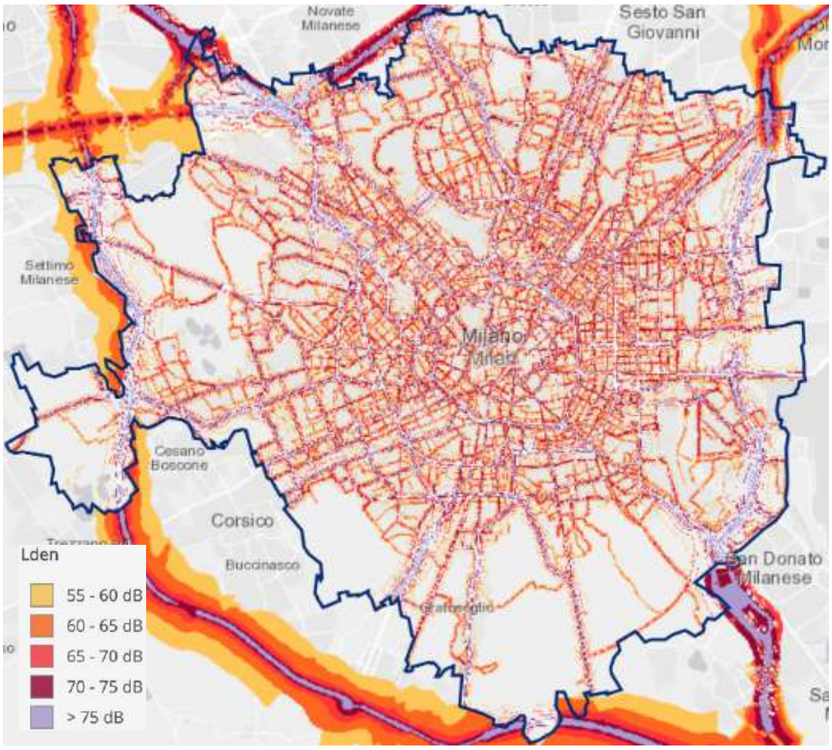


Figure 2.1: L_{den} map for road traffic noise in Milan. Source: http://noise.eea.europa.eu/

Moreover, EEA makes available data about the total number of people exposed to noise inside and outside urban areas. This data is shown in Table 2.1 and depicted in Figures 2.3 and 2.4.

The noise maps mentioned above and the data on the number of people exposed to noise levels over the threshold refer to the most recent country submissions of the 2017 round of noise reporting.

Table 2.1: Number of people exposed to L_{den} and L_{night} levels over the threshold. Source: https://www.eea.europa.eu/

Noise Source	$L_{den} \geq 55\ dB$	$L_{night} \geq 50\ dB$	Agglomeration	Country Group
Roads	78 236 200	55 088 700	Inside urban areas	EU-28
Railways	10 337 200	7 833 100	Inside urban areas	EU-28
Airports	3 028 200	857 400	Inside urban areas	EU-28
Industry	782 900	356 900	Inside urban areas	EU-28
Major roads	30 574 800	20 705 700	Outside urban areas	EU-28
Major railways	10 666 700	8826 700	Outside urban areas	EU-28
Major airports	1 072 800	439 800	Outside urban areas	EU-28
Roads	81 668 800	57 479 600	Inside urban areas	EEA-33
Railways	10 734 000	8 143 800	Inside urban areas	EEA-33
Airports	3 135 300	896 400	Inside urban areas	EEA-33
Industry	797 900	365 900	Inside urban areas	EEA-33
Major roads	31 142 900	21 096 500	Outside urban areas	EEA-33
Major railways	10 876 100	8994 100	Outside urban areas	EEA-33
Major airports	1 079 200	442 300	Outside urban areas	EEA-33

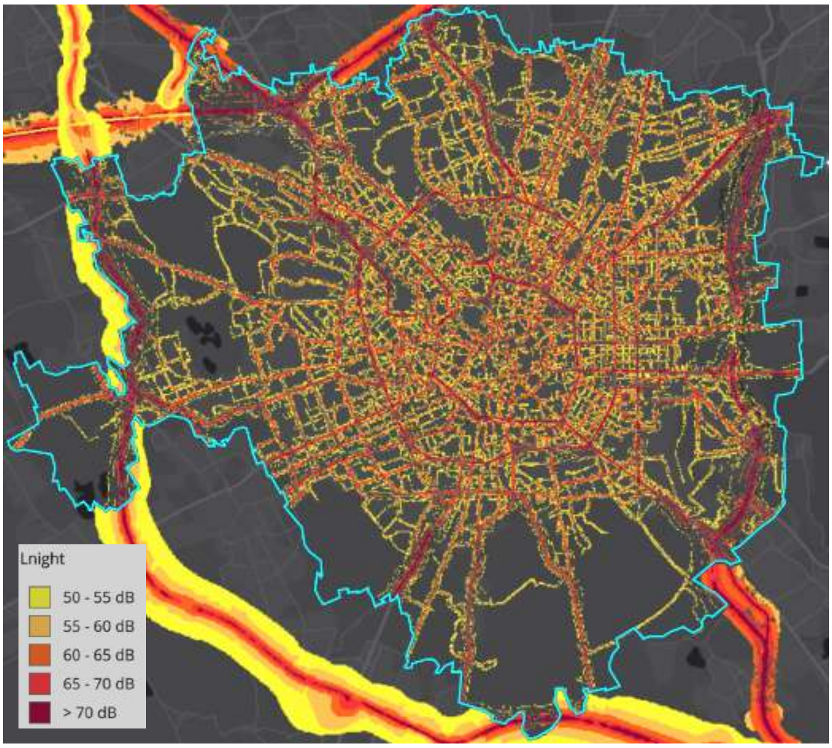


Figure 2.2: L_{night} map for road traffic noise in Milan.
Source: http://noise.eea.europa.eu/

2.2 Assessment Methods for the Noise Indicators

The END states that [1]:

> «the values of L_{den} and L_{night} can be determined either by computation or by measurement (at the assessment position). For predictions only computation is applicable»

In practise, however, the exposure values on noise maps, such as

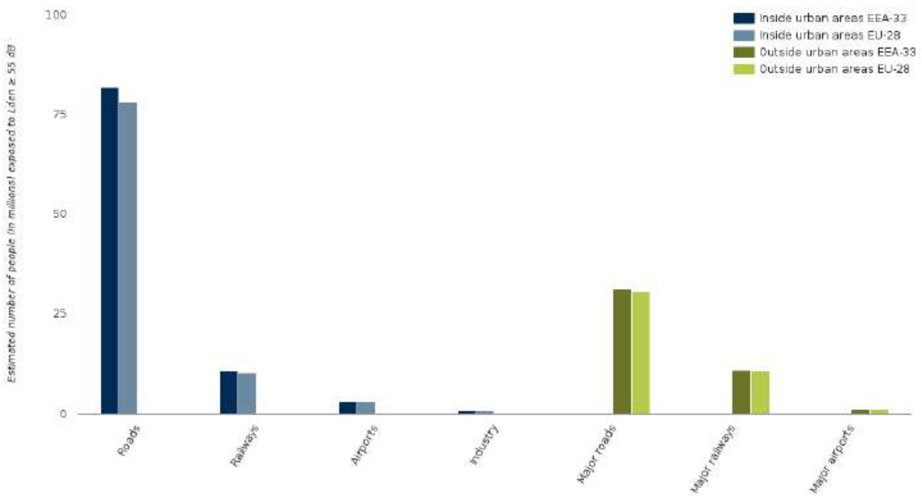


Figure 2.3: Number of people exposed to average day-evening-night noise levels $L_{den} \geq 55\ dB$ in Europe.

those in Figures 2.1 and 2.2, are always computed. The main reason is scalability: authorities have considered it infeasible to measure the sound level at all places and times. Hence, all END-compliant noise maps are produced with specialised software that simulates expected levels at different places and times. Such software uses source-specific sound propagation models that are fed with statistics about the presence of considered sources (e.g the average number of vehicles on roads, the frequency of planes on low-altitude flight paths, etc.), information on urban topology (e.g. the height of buildings, the width and surface type of roads, the presence of noise barriers, etc.) and weather conditions (e.g temperature, humidity, wind, etc.).
Another reason is the requirement to make separate noise maps per noise source: it is difficult to fulfil this requirement with measuring since sound level meters cannot differentiate between the sources.
The preference for computation was made explicit in the *Good Practice Guide for Strategic Noise Mapping and the Production of Associated Data on Noise Exposure* by the EC's *Working Group on the Assessment of Exposure to Noise* (WG-AEN) [6]:

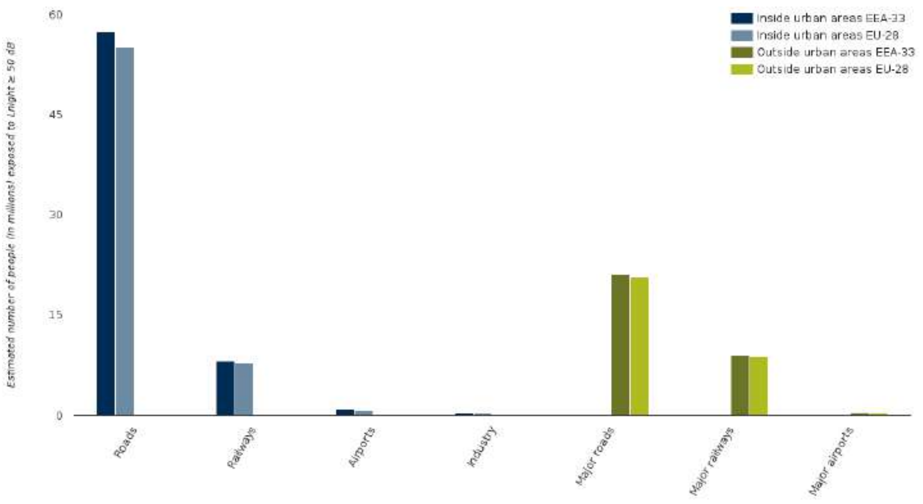


Figure 2.4: Number of people exposed to night noise levels $L_{night} \geq 50\ dB$ in Europe.

«WG-AEN recommends that wherever possible strategic noise mapping should generally be carried out by computation. However, it is recognised that noise measurement has many supplementary roles to play in the effective implementation of the END»

Two supplementary roles for measurements are the initialisation of models and the verification (and possibly the correction) of their output. Typically this only requires limited amounts of data which is either collected by designated officials equipped with professional-grade sound level meters, or by means of a sensor network.
The main advantage of simulation is that it allows to predict sound levels over huge areas with little measuring in the field , as long as statistics on considered sources are available.
The limitations of simulation are:

- **High cost** - Applying this method requires large datasets (which may not be publicly available), expensive software and a lot of human expertise

- **Uncertainty** - Because the output of the models is only validated with a limited amount of measurements, it is hard to estimate overall error margins. Moreover, accurate modelling of sound sources requires many different input parameters, all of which can influence the accuracy of the result

- **Limited amount of sources** - In reality the diversity of sources of potentially bothersome noise is much broader than the 4 sources considered by the END (road, rail and air traffic and industry). END-compliant maps ignore the noise produced by humans (e.g. noisy neighbours), construction sites and roadworks, passage of emergency services, various types of manifestations or events, etc. – all of which can be highly annoying to at least some city dwellers

- **Strategic focus** - Noise maps are valid for 5 years and represent expected sound levels for an average day or night. Hence they do not inform about incidental or short-term variations in exposure

Simulated maps are useful for authorities to understand the global trends in the urban soundscape, providing a lower limit on actual citizen exposure. They do not model local variations very well.

Another assessment method is based on sensor networks. These consist of nodes that are installed at various locations across a city and which autonomously and continuously measure the ambient sound level. Each node can be either stationary i.e permanently installed at the location or mobile i.e. installed on a vehicle. The nodes are distributed in such a way that the network covers different areas affected by different major sources of noise (e.g near roads, railways, etc.).
The main advantage of using sensor networks is that it allows to monitor the real-time sound level at certain places over arbitrary long periods of time, which is not feasible with manually operated equipment. The data that is collected with these networks is enables

to study temporal variations in the soundscape of specific places which is not possible with simulation-based noise maps because they lack a local, short-term perspective. Sensor network data can also be used to validate the output of simulation models.
The main limitation is the network sparsity. If the number of nodes is too low considering the size and population of the cities to monitor, the network is rather sparse causing reduce spatial coverage.

2.3 Real-Time Noise Mapping

As described in § 2.2, one limitation of simulation-based noise maps is that they can not inform about incidental or short-term variations in exposure. For instance, if road traffic is deviated due to roadworks this can significantly alter the soundscape of a neighbourhood for days, weeks or even months, but it will not be reflected in the maps. For this reason, research projects focused on development of systems suitable for real-time noise maps production. This is achieved by implementing a noise monitor network which continuously collects noise data, and transmits them to a data centre in which a noise mapping software is running. The mapping software computes noise maps according the information coming from the noise monitoring network.

Real-time noise mapping research projects have basically two different approach:

- **Participatory sensing**: this is a people-centric approach based on using smartphones as noise measurement devices thanks to applications that turn smartphone into geolocalized noise sensor. This approach enable citizens to measure their personal exposure to noise in their everyday environment and share measurement to produce collaborative noise maps. Examples are *NoiseTube* and *OpeNoise*

- **Non-Participatory sensing**: this approach is based on the implementation of a sensor network that consist of stationary nodes permanently installed at the location to be monitored. *Dynamap* is an example of non-participatory sensing.

2.3.1 Participatory Sensing: NoiseTube and OpeNoise

The assessment of data for the production of real-time noise maps needs a device equipped with:

- a microphone to acquire the noise signal
- hardware/software resources to elaborate that signal
- a GPS module to geolocalize the measurement
- a modem module to send the elaborated data to a data centre

Smartphones have all the features listed above. Hence, due to their growing popularity in the lasts years - smartphones have become a 'must have' for the majority of adult citizens in world's developed nations - research project focused on the realization of wireless sensor networks using smartphone as nodes.
Examples of this approach are *NoiseTube* [7] and *OpeNoise* [8]

NoiseTube is a research project started at the *Sony Computer Science Laboratory* in Paris in collaboration with *Vrije Universiteit Brussel*. Nowadays the project is maintained by the *Software Languages Lab* at *Vrije Universiteit Brussel*. The project objective is the development of a smartphone application to turn a mobile phone into a noise sensor. After installing the free NoiseTube application, the citizen is able to measure the A-weighted equivalent noise level (Figure 2.5a) and send this data to NoiseTube server where it is processed to produce collaborative noise maps (Figures 2.5b and 2.5c).

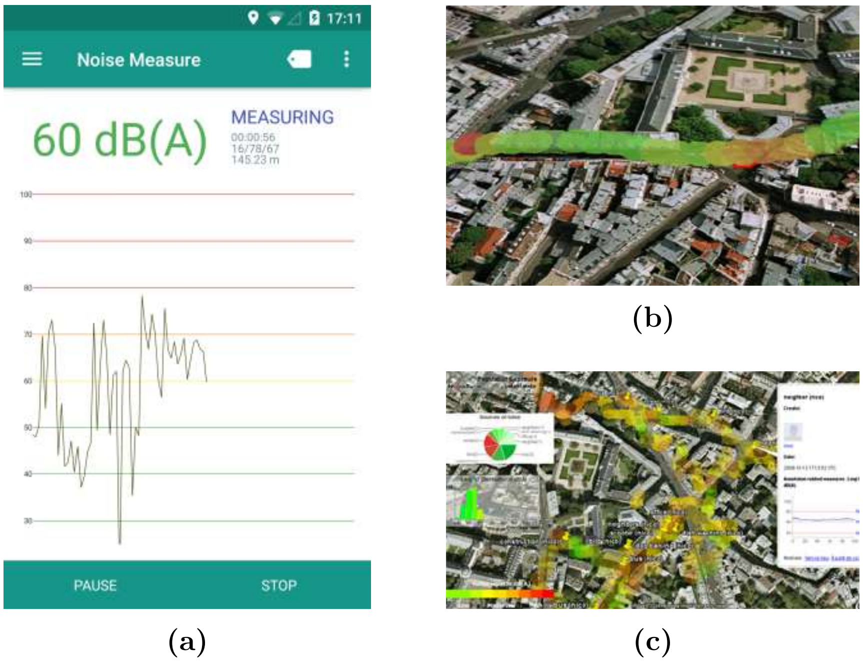


Figure 2.5: NoiseTube application: (a) L_{Aeq} mesurement; (b) the sound exposure of a user during a walk in Paris. Circles represent sound levels in $dB(A)$ as measured by the user; (c) map of noise pollution visualized in Google Earth. Source: http://www.noisetube.net

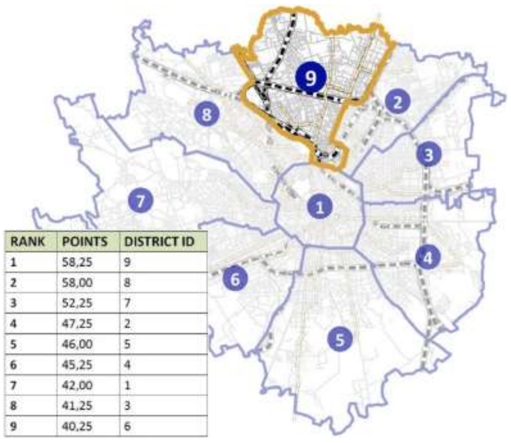
OpeNoise is another example of a smartphone application for noise measurement developed by *ARPA Piemonte*. Basically, OpeNoise has the same features of NoiseTube.

The main problem of smartphone-based participatory sensing is the measurement accuracy. In [9], 100 smartphones were tested using different sound level meter application . These measurement were compared with noise levels acquired with calibrated professional sound level meters. Tests were conducted on phones using the Android and iOS platforms: the results show that iOS applications have the best agreement with reference value. Android smartphones are less suitable for noise measurement because they are build by several different manufacturers. As a result, devices are build using different microphones and audio component. For this reason it is infeasible to use the same calibration algorithm for all the devices, making the measurements inaccurate.

2.3.2 Non-Participatory sensing: Dynamap

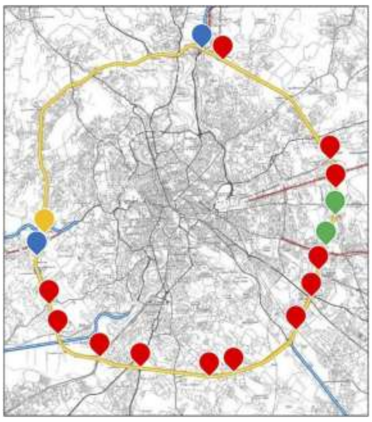
Dynamap [10] is a *LIFE+* project financed by European Commission. This project aims to develop a real-time noise maps for monitoring the acoustic impact of road infrastructure using a stationary nodes sensor network that consist of low-cost acoustical sensors. Dynamac is based on a non-participatory sensing approach as the citizens are not involved in the noise level measurement.
The Dynamap system is located in two pilot area: Milan district 9 (Figure 2.6a) and A90 Motorway surrounding the city of Rome 2.6b. These areas have been chosen to test the requirement associated to agglomeration and major roads as required by the END. The Milan pilot area cover a significant portion of the city including different types of roads and acoustical scenarios. The pilot area located in Rome allows to install noise sensors in hot spots for traffic so that it is possible to asses major road noise levels; such a sensor network uses 17 acoustical measurement devices. The collected noise data are sent to data centre. A software tool implemented on a general purpose geographic information system (GIS) platform elaborates these data to produce a noise map in real-time updated every 30 seconds. Real-time noise maps of the two pilot areas are available at Dynamap website[3][4] (Figure 2.7).

[3]https://milano.noisemote.com
[4]https://roma.noisemote.com



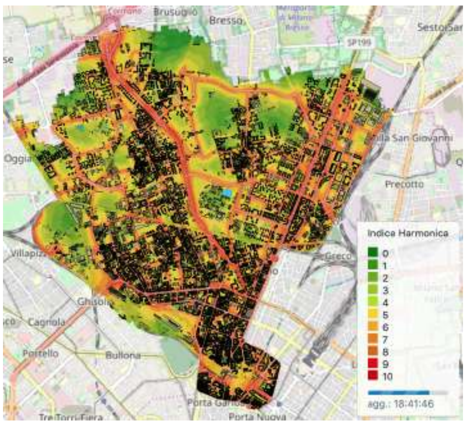
RANK	POINTS	DISTRICT ID
1	58,25	9
2	58,00	8
3	52,25	7
4	47,25	2
5	46,00	5
6	45,25	4
7	42,00	1
8	41,25	3
9	40,25	6

(a)

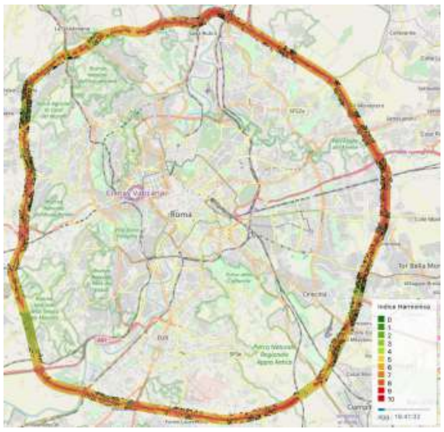


(b)

Figure 2.6: Dynamap project pilot areas: (a) Milan distric 9; (b) A90 Motorway (Rome). Source: [10].



(a)



(b)

Figure 2.7: Dynamap dynamic noise maps examples: (a) Milan. Source: https://milano.noisemote.com; (b) Rome. Source: https://roma.noisemote.com

Chapter 3

CAR-AUDIRE

CAR-AUDIRE is a project started by *Master of Sound and Entertainment Engineering* at the *Department of Electrical Engineering* at the *University of Rome Tor Vergata.* CAR-AUDIRE objective is to develop efficient, low cost, capillary and portable technologies suitable to assess environmental parameters in order to identify, locate and analyze the sources and locations of noise causing noise pollution or critical events for human health. The idea is based on the growing interest in both public and private sectors in making cities and territories more suitable for human well-being and safety. The project aims to using cars as distributed sensors able to make acoustic measurements that, sent by IoT wireless communication standards, are used to create real-time noise maps in cities as well as to recognize and react to sound events (e.g. car crashes). This is achieved by the integration of different types of advanced technologies applied to Automotive for the smart assessment of environmental noise - and more.
The project focus is to develop a device capable of measuring and analyzing environmental noise using low energy signal processing,

artificial intelligence techniques, GPS geolocation, within an Internet of Things (IoT) ecosystem and a Cloud services infrastructure, allowing great adaptability to a wide range of present and future applications. For instance, in addition to systematic environmental noise measures, it is possible to measure other environmental parameters simply adding the right sensors to the device to obtain measurements and maps of lighting, temperature, humidity, levels of pollutants, etc. Moreover, signal processing systems will be implemented allowing to analyze sound locally through machine learning techniques and artificial intelligence, in order to automatically identify critical sound events such as car crushes and explosions.
The device is energetically self-sufficient and does not require installation. Once it is placed on board, it automatically collects data, i.e. the driver does not need to start the measurement. The collected data are processed locally and sent anonymously in real-time to local authorities who are thus able on the one hand to immediately respond to emergencies, on the other to know the distribution on the territory of environmental parameters in order to make better and more informed decisions regarding safety and public health.

3.1 CAR-AUDIRE: Mobile Participatory sensing

CAR-AUDIRE project is based on mobile participatory sensing. The project introduces a new paradigm that allows to overcome the limitations of smartphone-based collaborative sensing.

The use of applications such as NoiseTube and OpeNoise is more likely by users with a personal interest in environmental noise. In other words, the average citizen is unlikely to use the smartphone as a noise measurement device because it is not its typical use. This can lead to a sparse data collection and spatial coverage. The use of vehicles makes everything more transparent: the user uses the car as

Table 3.1: Feaures Comparison: CAR-AUDIRE vs. project descrbed in § 2.3

Comparison Criterion	Smartphone-based Collaborative Sensing	Dynamap	CAR-AUDIRE
Spatial Coverage	N	N	Y
Mobile Installation	Y	N	Y
Fixed Installation	N	Y	Y
Accuracy	N	Y	Y
Real-Time Monitoring	N	Y	Y
Sound Event Detection	N	N	Y

usual (i.e. travel on the territory) and simultaneously collects data in a collaborative and completely "unconscious" way. Moreover, this feature the pervasive presence on the territory of cars increase the spatial coverage of the users and therefore of the measurements made on the territory.

CAR-AUDIRE overcomes the problem of device calibration because the sound level meters implemented in the devices have a uniform factory calibration.

Chapter 4

Sound Level Meter

A *Sound Level Meter* (SLM) is an instrument that measures sound pressure levels. It responds to sound in approximately the same way as the human ear and gives objective, reproducible measurements of sound pressure levels. It is designed to measure sound levels in a standardized way: the standard IEC 61672-1 [11] defines specification and quantities that a sound level meter must measure.

SLMs are used to measure and manage noise from a variety of sources. For example some applications in which a sound level meter is used are measures of environmental or neighbourhood sound, monitoring a live event, verifying a sound installation or the qualification of the machinery and industrial noise. The common aspect of all these applications is that we want to determine the noise emission to humans, so we want for instance to quantify the noise pollution or we want to identify annoying or hazard sound levels or optimizing a specific environment or installation. So it always has to do with the impact of sound on human hearing perception.

Figure 4.1 shows the components of a SLM. It is a combination of a microphone, a preamplifier, an amplifier, an analog to digital converter, a signal processor and a display device.

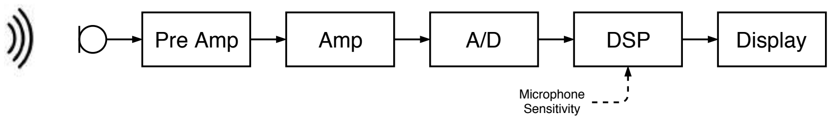


Figure 4.1: SLM Block Diagram.

4.1 Microphones

A microphone converts the variations in pressure of sound waves into time-varying electric signals. Ideally, the microphone should have the following characteristics [12]:

- The electric signal it generates should be an exact analog of the sound wave
- The presence of the microphone in a sound field should not disturb the sound field
- The microphone frequency response should be independent of frequency over a wide frequency range
- There should be a linear relation between the level of the output signal of the microphone and the level of the sound pressure at the microphone over a wide range of sound pressures and at all frequency within the useful range of the microphone
- The sensitivity of the microphone should not vary with time or with ambient conditions

In practice, microphones only approximate these ideal characteristics.

4.2 Amplifier

The amplifier in a SLM should meet the following requirements [12]:

- Amplify the signal from the microphone sufficiently to permit the measurement of low sound pressure levels
- Amplify sounds over a wide frequency range, usually between 1 and 10 Hz at the lower limit of nominally flat response, and above 20 kHz at the upper lit
- Generate a level of electrical noise within the instrument that is lower than the level of the lowest sound pressure signal to be measured at any frequency in the range of the microphone

4.3 Signal Processor

The signal processor consists of the functional blocks that calculate the quantities defined in the standard. Figure 4.2 shows the block diagram of the signal processor. As shown in Figure 4.1, the sound pressure level coming onto the front of the microphone is translated into a voltage signal by the microphone capsule. The voltage signal is then sampled an digitized and then the signal processor calculate the required results out of the digital data stream. An important side note is that the microphone sensitivity goes also into the calculation. The reason is that microphone sensitivity is a measurement of a microphone's efficiency as a transducer, i.e. how well it converts acoustical energy to electrical energy. Microphone sensitivity

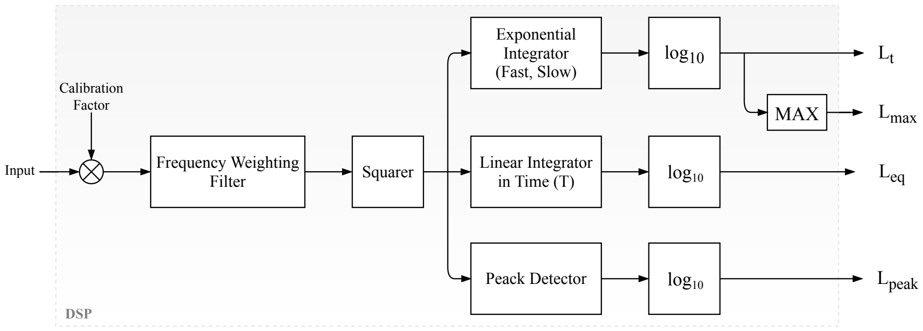


Figure 4.2: Signal Processor Block Diagram.

is determined by its output voltage relative to the sound pressure level it is subjected to. It is very important that microphone sensitivity goes into the calculation of the required results because we need the scaling so that the end result is accurately representing the physical situation in front of the microphone. For this reason, the sampled and digitized signal is multiplied by the *Calibration Factor* first, which is a multiplicative factor that take into account the microphone sensitivity.

4.3.1 Frequency Weighting Filters

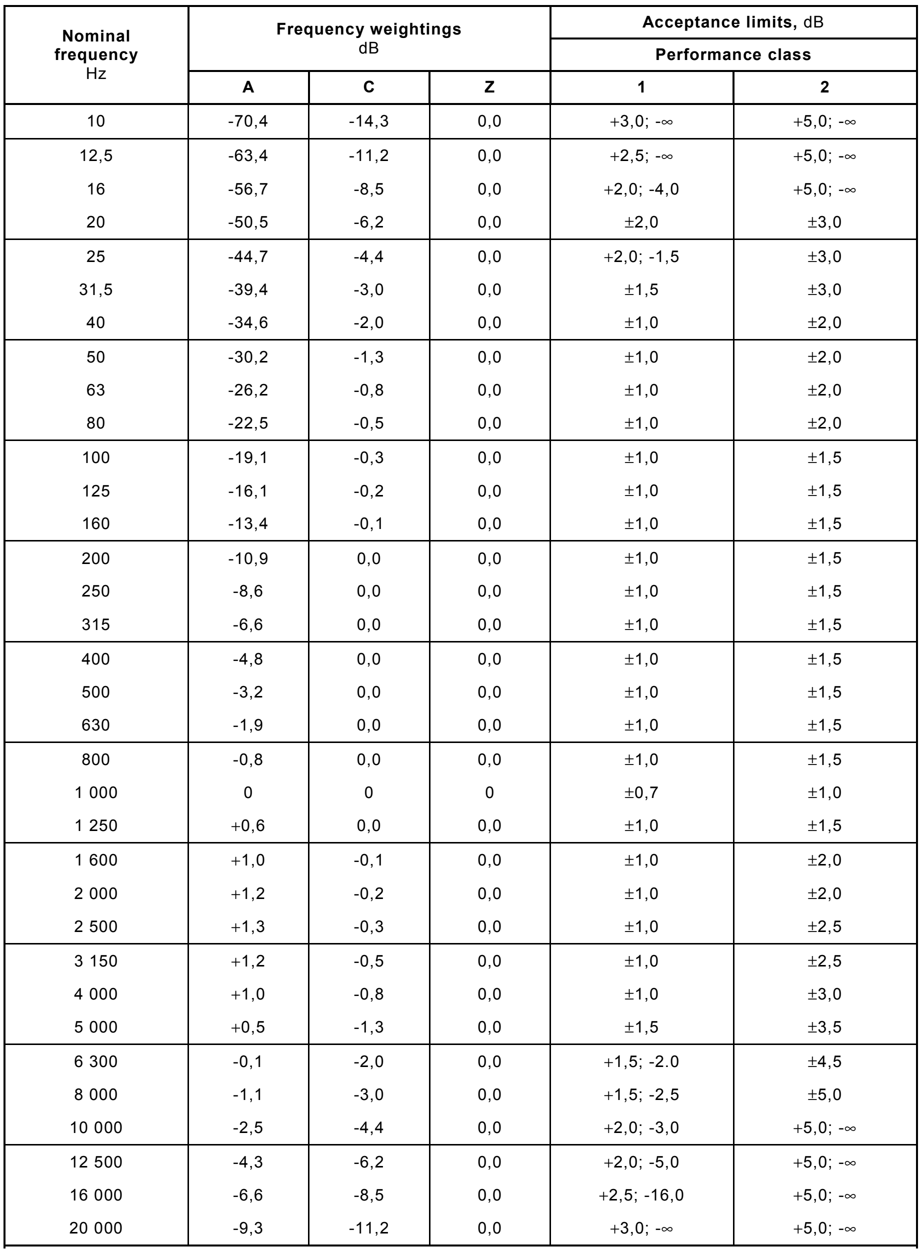
Human ear is not equally sensitive at all frequency. For this reason, even though the sound pressure level of two different sounds may be the same, the first may be judged louder than the second if the sound power of the first is concentrated in a frequency region where the human ear is more sensitive. Thus, sound pressure level is not a measure of the loudness of a sound. To obtain levels which have a closer relationship with loudness judgement than sound pressure levels, frequency weighting is incorporated in sound level meters to alter the sensitivity of sound level meter with respect to frequency so that the instrument is less sensitive at frequency where the human ear is less sensitive. This is achieved by using a *Frequency Weighting Filter*.

IEC 61672-1 defines frequency weightings A, C and Z giving a table showing the design-goals for for the three filters just mentioned (Figure 4.3) Figure 4.4 shows the filters' magnitude response. As we can see, for all frequency weighting the design goal includes a 0 dB weighting at 1 kHz.

A-weighting adjusts a signal in a way that resembles the human ear's response at medium-range levels. It is based on the 40 dB equal loudness curve. Its response decreases at frequency below 1 kHz so that the mid-frequencies and high frequencies are given greater emphasis(i.e. they are weighted less heavily).The A-weighting characteristic is consistent with the reduced sensitivity of normal human hearing at low frequencies compared with the response at higher frequencies [12]. A-weighting is required for nearly all environmental and workplace noise measurements and this kind of filters cover the full audio range, 10 Hz to 20 kHz.

C frequency weighting corresponds to the 100 dB equal loudness curve, that is the human ear's response at fairly high sound levels. Its response is fairly uniform from 50 Hz to 5 kHz C-weighting is mainly used when assessing peak values of high sound pressure levels. It can also be used, for example, for entertainment noise measurements, where the transmission of bass noise can be a problem.

'Zero' frequency weighting (no weighting and thus no filter) is a flat frequency response between 10 Hz and 20 kHz. It may be applied, for example, where an analysis of the sound source is required rather than the effect the sound has on humans, such as in testing the frequency response of produced loudspeakers in a manufacturing process.



Nominal frequency Hz	**Frequency weightings** dB			**Acceptance limits,** dB	
				Performance class	
	A	**C**	**Z**	**1**	**2**
10	-70,4	-14,3	0,0	+3,0; -∞	+5,0; -∞
12,5	-63,4	-11,2	0,0	+2,5; -∞	+5,0; -∞
16	-56,7	-8,5	0,0	+2,0; -4,0	+5,0; -∞
20	-50,5	-6,2	0,0	±2,0	±3,0
25	-44,7	-4,4	0,0	+2,0; -1,5	±3,0
31,5	-39,4	-3,0	0,0	±1,5	±3,0
40	-34,6	-2,0	0,0	±1,0	±2,0
50	-30,2	-1,3	0,0	±1,0	±2,0
63	-26,2	-0,8	0,0	±1,0	±2,0
80	-22,5	-0,5	0,0	±1,0	±2,0
100	-19,1	-0,3	0,0	±1,0	±1,5
125	-16,1	-0,2	0,0	±1,0	±1,5
160	-13,4	-0,1	0,0	±1,0	±1,5
200	-10,9	0,0	0,0	±1,0	±1,5
250	-8,6	0,0	0,0	±1,0	±1,5
315	-6,6	0,0	0,0	±1,0	±1,5
400	-4,8	0,0	0,0	±1,0	±1,5
500	-3,2	0,0	0,0	±1,0	±1,5
630	-1,9	0,0	0,0	±1,0	±1,5
800	-0,8	0,0	0,0	±1,0	±1,5
1 000	0	0	0	±0,7	±1,0
1 250	+0,6	0,0	0,0	±1,0	±1,5
1 600	+1,0	-0,1	0,0	±1,0	±2,0
2 000	+1,2	-0,2	0,0	±1,0	±2,0
2 500	+1,3	-0,3	0,0	±1,0	±2,5
3 150	+1,2	-0,5	0,0	±1,0	±2,5
4 000	+1,0	-0,8	0,0	±1,0	±3,0
5 000	+0,5	-1,3	0,0	±1,5	±3,5
6 300	-0,1	-2,0	0,0	+1,5; -2.0	±4,5
8 000	-1,1	-3,0	0,0	+1,5; -2,5	±5,0
10 000	-2,5	-4,4	0,0	+2,0; -3,0	+5,0; -∞
12 500	-4,3	-6,2	0,0	+2,0; -5,0	+5,0; -∞
16 000	-6,6	-8,5	0,0	+2,5; -16,0	+5,0; -∞
20 000	-9,3	-11,2	0,0	+3,0; -∞	+5,0; -∞

Figure 4.3: Frequency Weightings and Acceptance Limits. Source [11].

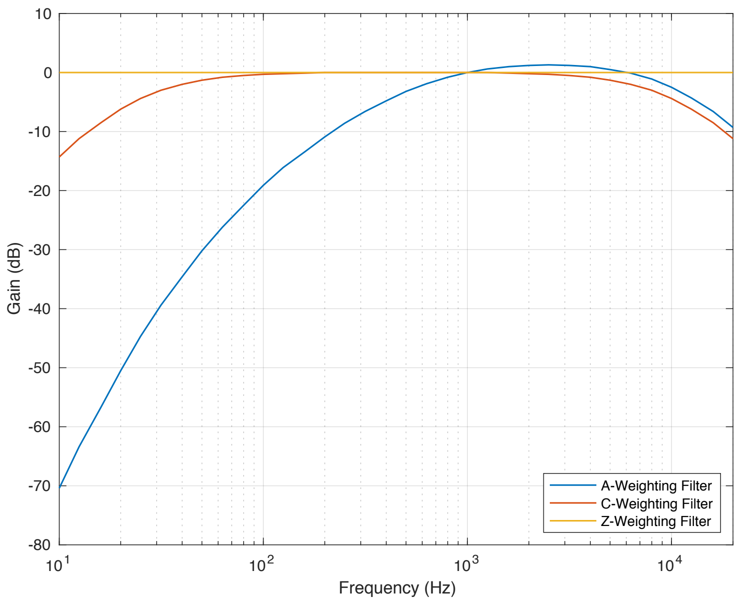


Figure 4.4: A-, C-, Z-Weighting Filters Magnitude Response.

4.3.2 Exponential Integrator: Time-Weighted Sound Level

SLMs were initially developed in the 1930s with a dial-and-pointer display i.e. a needle pointer moved across a calibrated scale indicating the sound level in decibels. Depending on the characteristics of the sound being measured, the fluctuations could be so rapid to make it difficult to determine an average position of the pointer. A measurement of sound level so obtained was termed a *fast* sound level. to reduce the variability in the measurements, electrical damping was added. A measurement of sound level with this additional damping was termed a *slow* sound level. Then, Time weighting specifies how the SLM reacts to changes in sound pressure.

The standard IEC 61672-1 define time-weighted sound level L_t as ten times the logarithm to the base 10 of the ratio of the running time average of the time- weighted square of a frequency-weighted sound-pressure signal to the square of the reference value

$$L_t(t) = 10\log_{10}\left[\frac{\frac{1}{\tau}\int_{-\infty}^{t} p^2(\xi)e^{-(t-\xi)/\tau}d\xi}{p_0^2}\right] \tag{4.1}$$

where

- τ is the exponential time constant in seconds
- ξ is a dummy variable of time integration from some time in the past, as indicated by $-\infty$ for the lower limit of the integral, to the time of observation t
- $p(\xi)$ is the frequency weighted instantaneous sound pressure signal
- p_0 is the reference pressure value of 20 μPa which is approximately the human hearing threshold at 1 kHz

Time-weighted sound level is expressed in decibels (dB).
For *fast*, the exponential time constant is $\tau_F = 0.125$ seconds; for *slow*, it is $\tau_S = 1$ second.
The choice of exponential-time-weighting for use in a particular measurement depends on the variability of the sound signal and the requirements of the applicable measurements standard.

The process indicated by Eq. 4.7 involves the following steps:

1. Start with a frequency-weighted electrical input signal
2. Square the input signal
3. Apply a low-pass filter with one real pole at $-\frac{1}{\tau}$
4. Take the base-10 logarithm

Figure 4.5 shows how the exponential weighting influences the weight of the samples involved in the computation of the time-weighted sound level for a particular 2 second of a non-steady sound. The independent variable for the graph is the relative time before the time of observation at time t. Figure 4.5a and b shows the instantaneous A-weighted sound pressure and the square of the instantaneous A-weighted sound pressure respectively. The standard exponential-time-weightings are shown in Figure 4.5c at the time of observation. As the running time variable ξ increases from the start time to the observation time t, the exponential-time-weighting $e^{-(t-\xi)/\tau}$ increases exponentially to 1. Figure 4.5d and e shows the result of multiplication of the squared A-weighted sound pressure in Figure 4.5b by the exponential-time-weightings in Figure 4.5c. As we can se, in the case of *fast* exponential-time-weighting only the last sample has a weight of 1 whereas the other samples have a rapidly declining weights, so only the sample within the last second play a role in the computation of L_t. In other words, the *fast* result is only a glimpse of the very last sound pressure that has happened during the measure. The *slow* exponential-time-weighting has a slower decay so there are more samples going in the computation of the *slow* result. In this way we achieve a more smoothed measure display.

Figure 4.6 shows the time variation of *fast* and *slow* time-weighted sound level.

In relation to the time-weighted sound level L_t, the standard IEC 61672-1 defines the maximum time-weighted sound level L_{max} as the greatest time-weighted sound level within a stated time interval T [11]

$$L_{max} = \max_{L_t \in T} [L_t(t)] \tag{4.2}$$

where

- $L_t(t)$ is the time-weighted sound level
- T is the measurement time interval

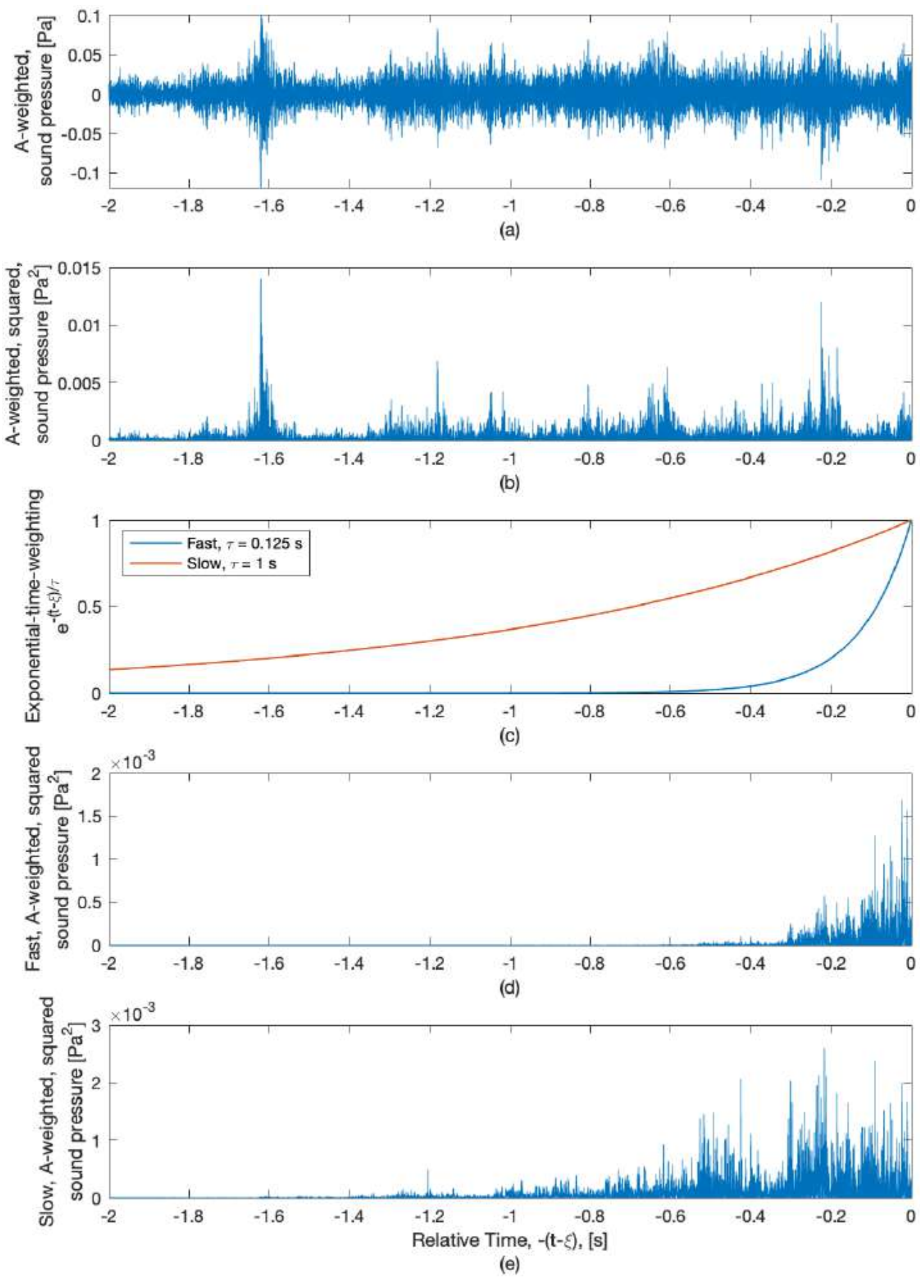


Figure 4.5: Exponential weighting influence on the L_t computation. (a) The instantaneous A-weighted sound pressure; (b) the square of the instantaneous A-weighted sound pressure; (c) exponential-time-weightings at the time of observation; (d) and (e) the result of multiplying the A-weighted, squared sound pressure by the *fast* and *slow* exponential-time-weightings.

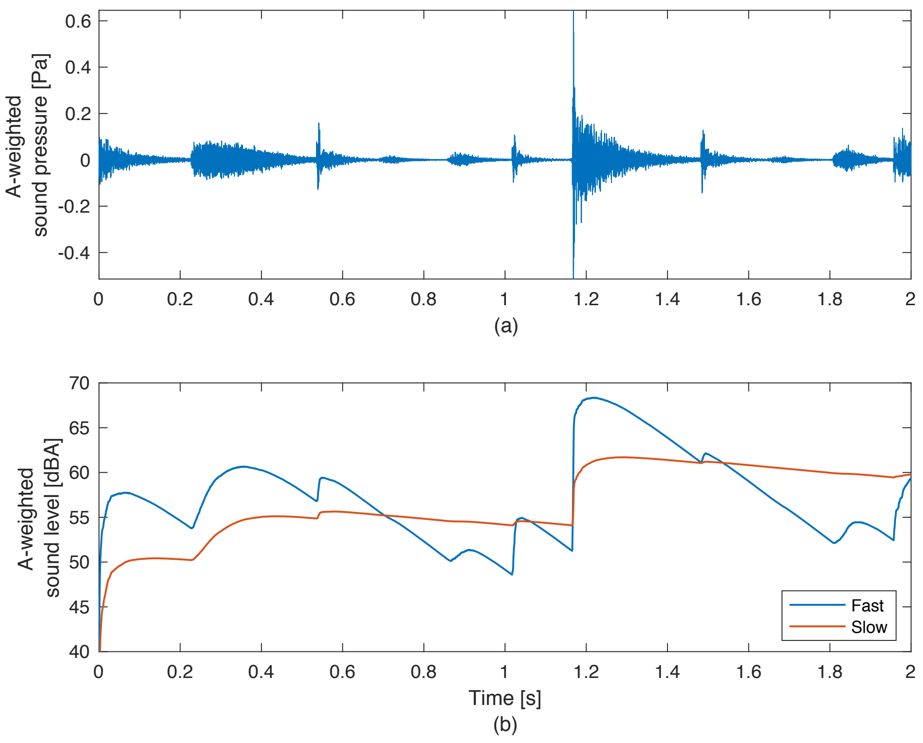


Figure 4.6: Time variation of *fast* and *slow* time-weighted sound level.

Maximum time-weighted sound level is expressed in decibels (dB). Figure 4.7 shows the maximum-time-weighted sound level of the fast time-weighted sound level shown in Figure 4.6 using two different measurements period.

4.3.3 Linear Integrator: Equivalent Continous Sound Level

The instantaneous sound pressure level present in front of the microphone usually changes pretty rapidly. As it has high fluctuations,

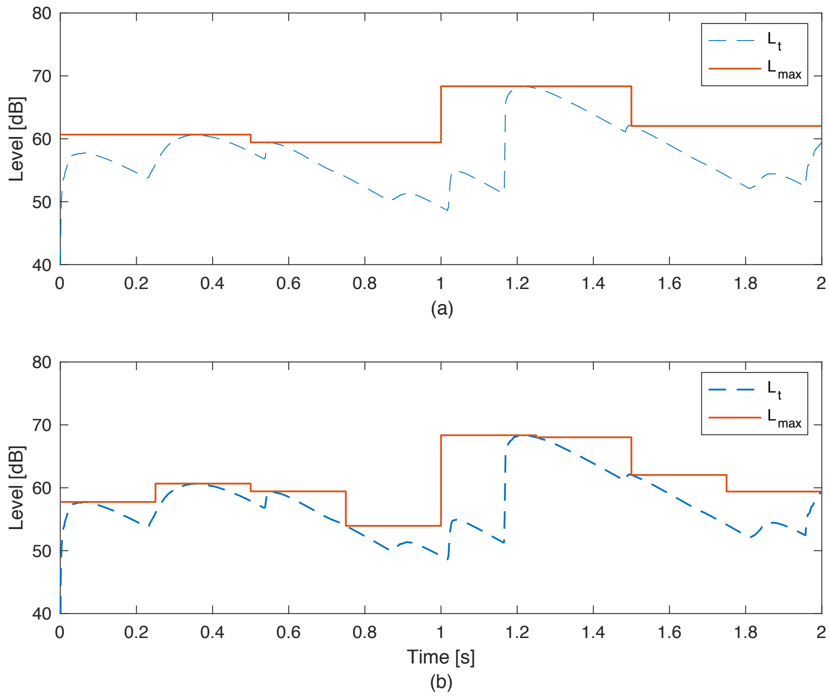


Figure 4.7: Comparison of two L_max measure using two different measurements time T. (a) Maximum-time-weighted sound level for measurement time $T = 0.5\ s$; (b) maximum-time-weighted sound level for measurement time $T = 0.25\ s$

it would be difficult to get an overall picture of an event by just watching the instantaneous sound pressure level. For that reason, in most application we are using *equivalent continuous sound level* L_{eq}.

Equivalent continuous sound level is the constant sound pressure level that would produce the same energy as the fluctuating sound level over a given time interval T.

It has a very good correlation whit human perception and its value would give us a good picture of what a human being having been present at an event would has perceived of the loudness. So L_{eq} is the most relevant measure to correlate to the human hearing perception.

Equivalent continuous (or time-averaged) sound level is defined in [11] as ten times the logarithm to the base 10 of the ratio of the time average of the square of a frequency-weighted sound-pressure signal during a stated time interval to the square of the reference value

$$L_{eq} = 10 \log_{10} \left[\frac{\frac{1}{T} \int_{t-T}^{t} p^2(\xi) d\xi}{p_0^2} \right] \tag{4.3}$$

where

- ξ is a dummy variable of time integration over the averaging time interval ending at the time of observation t
- T is the averaging time interval
- $p(\xi)$ is the frequency-weighted sound pressure signal
- p_0 is the reference pressure value of 20 μPa which is approximately the human hearing threshold at 1 kHz

Equivalent continuous sound level is expressed in decibels (dB). In principle, time weighting is not involved in a determination of equivalent continuous sound level.

In the equivalent continuous sound level we have an integral so that means that we have averaging the instantaneous sound level over the measuring period T. Whereas the instantaneous sound level gives as an information about the instantaneous status, L_{eq} gives us the information of the average level at the end of the measuring period T.

L_{eq}, like L_t and L_{max}, is a RMS (Root Mean Square) value, which is a measure for the power of the incoming signal. In fact we can define

L_{eq} also as ten times the logarithm to the base 10 of the ratio to the square of the RMS value of the frequency-weighted sound-pressure signal during a stated time interval to the reference pressure value

$$L_{eq} = 10\log_{10}\left(\frac{p_{RMS}}{p_0}\right)^2 \tag{4.4}$$

Since

$$p_{RMS} = \sqrt{\frac{1}{T}\int_{t-T}^{t} p^2(\xi)d\xi} \tag{4.5}$$

Using Eqs. 4.4 and 4.5 we can write

$$\begin{aligned} L_{eq} &= 10\log\left(\frac{\sqrt{\frac{1}{T}\int_{t-T}^{t} p^2(\xi)d\xi}}{p_0}\right)^2 \\ &= 10\log\left[\frac{\left(\sqrt{\frac{1}{T}\int_{t-T}^{t} p^2(\xi)d\xi}\right)^2}{p_0^2}\right] \\ &= 10\log\left[\frac{\frac{1}{T}\int_{t-T}^{t} p^2(\xi)d\xi}{p_0^2}\right] \end{aligned} \tag{4.6}$$

which is the same expression as the standard definition of Eq. 4.3.

Figure 4.8 shows an application of the use of two different measurement time intervals for a particular two seconds of non-steady sound. The A-weighted measured sound pressure is shown in Figure 4.8a and equivalent continuous sound levels are shown in Figure 4.8 for measurement times of 0.5 second and 0.25 second.

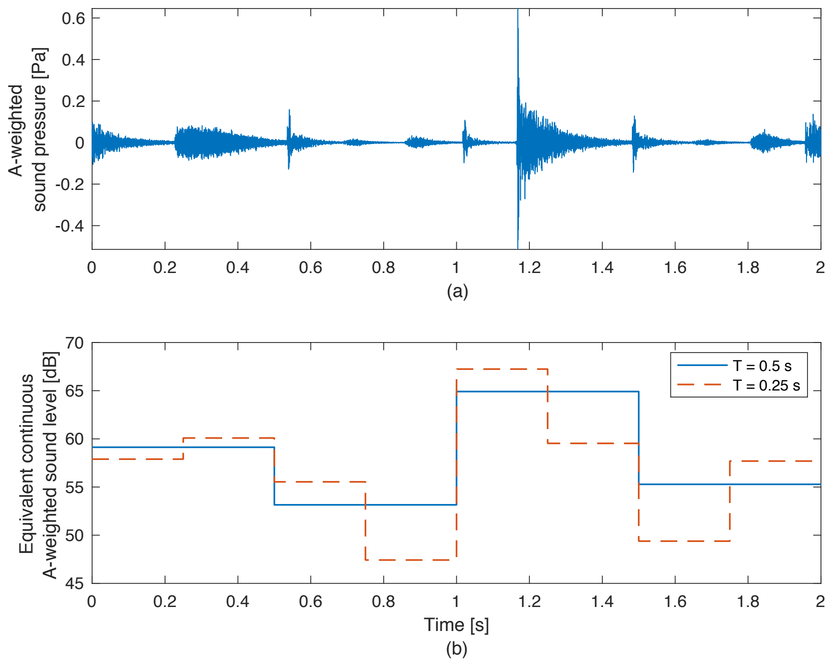


Figure 4.8: Comparison of two L_{eq} measures using two different measurements period T. (a) Waveform of A-weighted sound pressure; (b) Equivalent continuous sound levels for measurement times of 0.5 second and 0.25 second.

4.3.4 Peak Detector: Peak Sound Level

Peak detector captures the highest instantaneous frequency weighted sound pressure that occurs in a given measurement period T

$$p_{peak} = \max_{p(t) \in T} \left[p^2(t) \right] \tag{4.7}$$

where

- $p(t)$ is the instantaneous frequency weighted sound pressure
- T is the measurement period

Peak sound level L_{peak} is defined as ten times the logarithm to the base 10 of the ratio of the square of a frequency-weighted peak sound-pressure signal to the square of the reference value [11]

$$L_{peak} = 10 \log_{10} \left[\frac{p_{peak}}{p_0^2} \right] \tag{4.8}$$

where

- p_{peak} is defined by Eq. 4.7
- p_0 is the reference pressure value of 20 μPa which is approximately the human hearing threshold at 1 kHz

Peak sound level is expressed in decibels (dB).

As shown in Figure 4.9, peak sound level is not the same as maximum sound level. In fact unlike L_{max}, the L_{peak} computation does not involves any time weighting and it is not a RMS value.

The L_{peak} is used for noise measurement where loud bangs are present; it is not usually used for environmental noise measurement and is useless when any wind is present. A gust of wind will easily give very high L_{peak} readings.

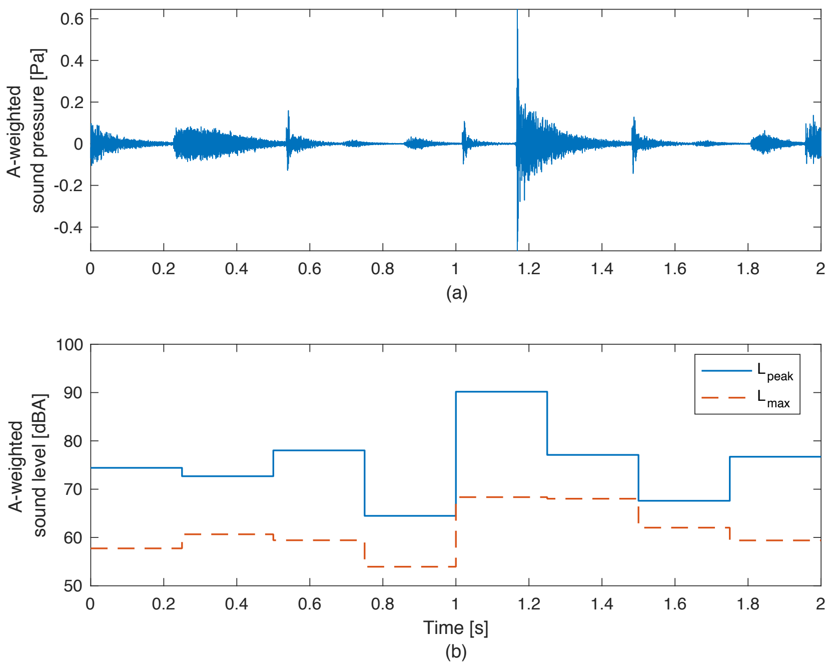


Figure 4.9: Comparison between L_{peak} and L_{max}. (a) Waveform of A-weighted sound pressure; (b) Peak sound level and maximum sound level for a measurement period $T = 0.25\ s$.

Chapter 5

Weighting Filters

This chapter describes how to implement digital frequency-weighting and time-weighting filters according to the specifications given by the standard [11].

5.1 Frequency Weighting Filter

Frequency weighting filters are required to correlate the physical noise sound pressure level measurements to the human's response to an auditory stimulus.

IEC 61672-1, in addition to the frequency weighting and acceptance limits shown in Figure 4.3, gives the equation required to calculate the analog (s-domain) frequency response of both A- and C-weightings.

Equation 5.1 shows the frequency response expression for a C-weighting filter

$$H_C(f) = k_C \frac{f_4^2 f^2}{(f^2 + f_1^2)\,(f^2 + f_4^2)} \tag{5.1}$$

where

- $f_1 = 20.6\ Hz$
- $f_4 = 12194\ Hz$
- $k_C = 1.0072$ is a constant value used to normalize the function to a gain of 1 (0 dB) at 1000 Hz

This is a filter with the following characteristics:

- Two zeros
- Two poles at $-20.6\ Hz$
- Two poles at $-12194\ Hz$

and therefore has the following transfer function

$$H_C(s) = G_C \frac{\omega_4^2 s^2}{(s + \omega_1)^2\,(s + \omega_4)^2} \tag{5.2}$$

Adding two real-axis poles to the C-weighting transfer function gives us the A-weighting transfer function.
Equation 5.3 shows the frequency response expression for an A-weighting filter

$$H_A(f) = k_A \frac{f_4^2 f^4}{(f^2 + f_1^2)\,(f^2 + f_2^2)^{1/2}\,(f^2 + f_3^2)^{1/2}\,(f^2 + f_4^2)} \tag{5.3}$$

where

- $f_1 = 20.6\ Hz$
- $f_2 = 107.7\ Hz$
- $f_3 = 737.9\ Hz$
- $f_4 = 12194\ Hz$
- $k_A = 1.2589$ is a constant value used to normalize the function to a gain of 1 (0 dB) at 1000 Hz

This is a filter with the following characteristics:

- Four zeros
- Two poles at -20.6 Hz
- One pole at -107.7 Hz
- One pole at -737.9 Hz
- Two poles at -12194 Hz

and therefore has the following transfer function

$$H_A(s) = G_A \frac{\omega_4^2 s^4}{(s+\omega_1)^2 (s+\omega_2)(s+\omega_3)(s+\omega_4)^2} \tag{5.4}$$

The normalization constant are defined in [11] as $G_C = 0.062\ dB$ and $G_A = -2\ dB$.

IEC 61672-1 just describes suitable weighting filters giving Equations 5.1 an 5.3 but do not explain how to implement them for digital recorded sound pressure level data. We have to apply a method for deriving a digital filter from an analogue s-domain filter.

There are essentially two methods of filtering a signal digitally: Finite Impulse Response (FIR) and Infinite Impulse Response (IIR). Whilst FIR filters have some advantages (for example they are always stable and they can be design to have linear phase), their main disadvantages is that they would need a large number of coefficient to match the frequency magnitude response of the weighting filters given in the standard.
For these reason, we implemented the weighting filters using an IIR filter.

There are several method for deriving digital filter from an anlog s-domain filter. Two of these methods have proven to be useful for most application: Impulse Invariance and Bilinear Transform. Only the bilinear transform provides a general-purpose conversion function that can be used for lowpass, highpass, bandpass. and bandstop responses. The impulse invariant and step invariant conversion functions are quite difficult to apply and can only be used for lowpass filters [13] For these reasons, we apply the bilinear transformation method for deriving the digital filter from the analog s-domain filter.

Bilinear Transformation is a transformation between the variables s and z that maps the entire $j\Omega$-axis in the s-plane to one revolution of the unit circle in the z-plane. It correspond to replacing continuous-time variable s by

$$s = \frac{2}{T}\left(\frac{1 - z^{-1}}{1 + z^{-1}}\right) \tag{5.5}$$

where T is the sampling period. Since $-\infty \leq \Omega \leq \infty$ maps onto $-\pi \leq \omega \leq \pi$, there is a non-linear relationship between the analog frequency and the digital frequency, that is the frequency response will be distorted. Pre-warping the frequencies used in the analog s-domain equation (with the substitution shown in Equation 5.6) can eliminate the problem

$$\omega_n^{'} = \frac{2}{T}\tan\left(\frac{\omega_c}{2}\right) \tag{5.6}$$

Since A-weighting is required for nearly all environmental noise mea-

surements, we focused on implementing this filter.
We implemented the A-weighting filter using a cascade of second order filters, or biquads. Because of coefficient sensitivities in higher order IIR filters, the biquad is often used as the basic building block for more complex filters.

A biquad filter has the following transfer function in the z-domain

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2}} \tag{5.7}$$

A second-order section digital filter is specified as an $N \times 6$ matrix named *SOS*, where N is the number of second order section wehre E each row of the matrix contains the numerator and denominator coefficients for each filter. We need a cascade of three second order filters to match the frequency magnitude response of the A-weighting filter given in the standard. We have the following coefficient

$$SOS_A = \begin{bmatrix} 0.2557 & 0.5115 & 0.2557 & 1 & -0.1405 & 0.0049 \\ 1 & -2 & 1 & 1 & -1.8849 & 0.8864 \\ 1 & -2 & 1 & 1 & -1.9941 & 0.9941 \end{bmatrix} \tag{5.8}$$

5.2 Time Weighting Filter

The time weighting filter is involved in the computation of the time-weighted sound level (§ 4.3.2). The term $\frac{1}{\tau}\int_{-\infty}^{t} p^2(\xi)e^{-(t-\xi)/\tau}$ of Equation 4.7 as the convolution between the square frequency weighted sound pressure $p^2(t)$ with a filter with an impulse response

$$h(t) = \frac{1}{\tau}\, e^{-\frac{t}{\tau}} \tag{5.9}$$

This impulse response corresponds to a lowpass filter of the form

$$H(s) = \frac{\frac{1}{\tau}}{s + \frac{1}{\tau}} \tag{5.10}$$

It can be converted into the digital equivalent using impulse invariant method.

The impulse invariance method for transforming continuous-time filters into discrete-time filters, the impulse response of the digital filter is chosen proportional to equally spaced samples of the analog filter.

The impulse response of the discrete-time filter obtained by sampling $h(t)$ is

$$h[n] = Th(nT) = T\frac{1}{\tau}e^{-\frac{nT}{\tau}} = T\frac{1}{\tau}\left(e^{-\frac{T}{\tau}}\right)^n \tag{5.11}$$

where T is the sampling period.
The transfer function in the z-domain is given by

$$H(z) = \frac{\frac{1}{\tau f_s}}{1 - e^{-\frac{1}{\tau f_s}} z^{-1}} \tag{5.12}$$

Chapter 6

System Implementation

The system is implemented on a Commercial Off The Shelf (COTS) hardware allowing a fast prototyping approach to test the measurement of data and their transmission to the database. The available technology is already highly aggressive and tested. The IoT platform is based on a very compact COTS system consisting of a GPS module, NB-IoT (Narrow Band IoT) module and a low power microprocessor. This platforms is designed to be powered by *energy harvesting* methodologies. Since the power consumption is very low it should allow the implementation of an energy self-sufficient IoT node with the advantage of not being intrusive in the installation on cars. We have selected the SODAQ SARA AFF R410M developer board.

For future technological evolution of the system as a system on chip (SoC), we evaluated the possibility of using a RISC-V Instruction Set Architecture (ISA). RISC-V is an open-source hardware ISA based on established reduced instruction set computer (RISC) principles. It is increasingly drawing the attention of both industry and academia because it offers the possibility to ease the design of processors from the high costs of compiler support, without requiring to resort to expensive commercial ISAs. In particular, RISC-V is an interesting solution for IoT: the flexibility, extensibility and scalability of the RISC-V ISA has ushered in new possibilities for the IoT, making it easier to design IoT hardware and software solutions. The study we carried out about RISC-V is shown in Appendix A.

6.1 SODAQ SARA AFF R410M

SODAQ SARA Arduino Form Factor (AFF) R410M is an IoT developer board that allows to utilize NB-IoT networks with integrated GPS, Accelerometer and Magnetometers, Grove Connectors, JST connectors. The board is equipped with the Microchip SAMD21 microcontroller. It can be powered by battery or energy harvesting methodologies (solar). The basic moduls for system implementation are:

- Microchip SAMD21 microcontroller
- Ublox M8Q GPS/GNSS module
- Ublox SARA NB-IoT module

6.1.1 Microchip SAMD21 Microcontroller

The board is equipped with the ATSAMD21J18 microcontroller. This is a SAMD21 series microcontroller with 64 pins and a flash memory size of 256KB using the 32-bit ARM Cortex-M0+ processor. The microcontroller can operate at a maximum frequency of 48 MHz. The SAM D21 provide the following features: In-system

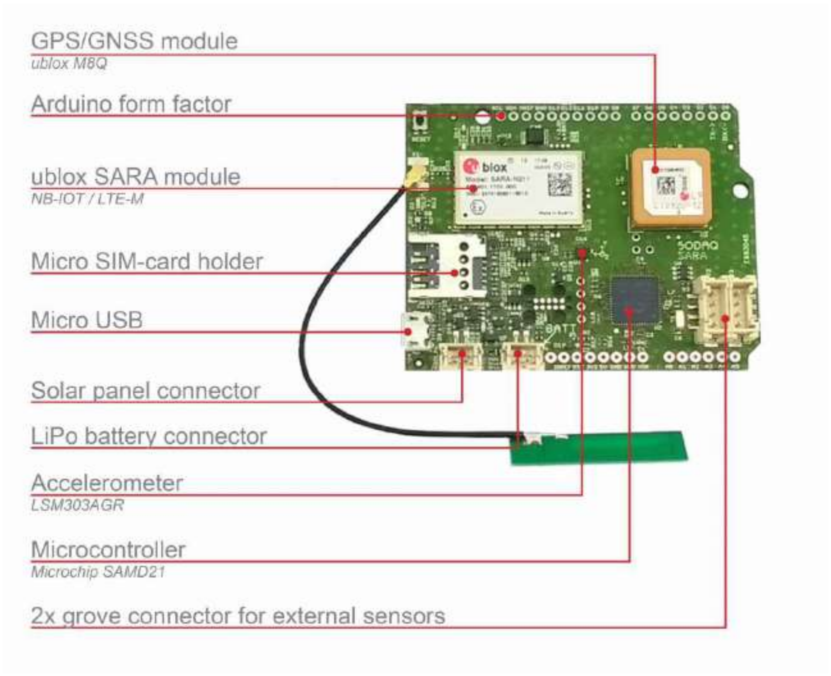


Figure 6.1: SODAQ SARA AFF R410M

programmable Flash, 12-channel Direct Memory Access Controller (DMAC), 12-channel Event System, programmable Interrupt Controller, 52 programmable I/O pins, 32-bit Real-Time Clock and Calendar (RTC), three 16-bit Timer/Counters (TC) and three 24-bit Timer/Counters for Control (TCC), where each TC can be configured to perform frequency and waveform generation, accurate program execution timing or input capture with time and frequency measurement of digital signals. The TCs can operate in 8- or 16-bit mode, selected TCs can be cascaded to form a 32-bit TC, and three timer/counters have extended functions optimized for motor, lighting, and other control applications. It provide one full-speed USB 2.0 embedded host and device interface; six Serial Communication Modules (SERCOM) that each can be configured to act as an USART, UART, SPI, I2C up to 3.4MHz, SMBus, PMBus, and LIN slave; two-channel I2S interface; a twenty-channel 350 ksps 12-bit ADC with programmable gain and optional oversampling and

decimation supporting up to 16-bit resolution, one 10-bit 350 ksps DAC, two analog comparators with Window mode, Peripheral Touch Controller (PTG) supporting up to 256 buttons, sliders, wheels, and proximity sensing; programmable Watchdog Timer (WDT), brown-out detector and power-on Reset and two-pin Serial Wire Debug (SWD) program and debug interface.
The microcontroller has accurate and low-power external and internal oscillators. All oscillators can be used as a source for the system clock. Different clock domains can be independently configured to run at different frequencies, enabling power saving by running each peripheral at its optimal clock frequency, and thus maintaining a high CPU frequency while reducing power consumption.
The SAM D21 have two software-selectable sleep modes, Idle and Stand-by. In Idle mode, the CPU is stopped while all other functions can be kept running. In Stand-by mode, all clocks and functions are stopped, expect those selected to continue running. The device supports SleepWalking. This feature allows the peripheral to wake up from sleep based on predefined conditions, and thus allows the CPU to wake up only when needed, e.g., when a threshold is crossed or a result is ready. The Event System supports synchronous and asynchronous events, allowing peripherals to receive, react to and send events even in Stand-by mode.

6.1.2 Ublox M8Q GPS/GNSS Module

The Ublox SAM-M8Q is a concurrent GNSS patch antenna module that offers high sensitivity and minimal acquisition times in an ultra-compact form factor. It can receive and track multiple GNSS systems: GPS, Galileo and GLONASS. Owing to the dual-frequency RF front-end architecture, GLONASS can be processed concurrently with GPS and Galileo signals, thus providing reception of three GNSS systems. By default, the M8 receiver is configured for concurrent GPS and GLONASS. If power consumption is a key factor, then the receiver should be configured for a single GNSS operation using GPS, Galileo or GLONASS. The module can also be configured to receive any single GNSS constellation.

6.1.3 Ublox SARA NB-IoT Module

Ublox SARA-R4 module provide Narrow Band IoT radio access technology. It is an optimal choice for low-powered wide area (LPWA) networks applications with low to medium data throughput rates, as well as devices that require long battery lifetimes, such as used in IoT applications.

6.2 Sensors

SODAQ recommends the Grove - Loudness sensor by Seeed Studio for noise measurements and sound detection. It is based on the

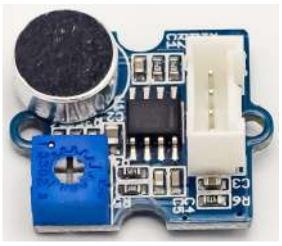


Figure 6.2: Groove Loudness sensor by Seeed Studio

LM2904 amplifier and a built-in electret microphone. Moreover, there is a screw potentiometer that enables manual adjustments to the output gain.

Chapter 7

Power Consumption Measurement

Currently, an experimentation phase has begun for the CAR AUDIRE project, developed in a series of measures of dynamic power consumption with dedicated instrumentation. To evaluate the system power consumption and the energy required to perform a measurement and data transmission operation, Table 7 shows the current consumption datasheet VALUES of the Ublox SARA N211, Ublox SAM M8Q and ATSAMD21J18 modules.

The measuring instrument used for power consumption measurment is the *Keysight N6705C DC Power Analyzer* (Figure 7.1). The N6705C is an interesting instrument designed to increase the productivity of the R&D engineer. It does that by combining multiple instrument function into a single package. In this instrument we can

Table 7.1: Current consumption of ATSAMD21J18, Ublox SARA R4 and Ublox SAM M8Q and modules

Module	Low Power Consumption	Working
ATSAMD21J18	~2 mA (Idle mode) 53.3 μA (Stand-by mode)	6.47 mA
Ublox SARA R4	$< 3\mu A$ (Deep-Sleep Mode)	< 220 mA (Tx)
Ublox SAM M8Q		29 mA

have up to four DC power supply, a voltmeter and an ammeter, an oscilloscope, a data logger and an arbitrary waveform generator.
The instrument has a fully integrated voltmeter and ammeter to measure the actual voltage and current being sourced out of the DC output into the device under test (DUT). Because this voltmeter/ammeter function is built in, it is easy to make measurements without additional wires or the added complexity of current sense resistors or current shunts. Using the data log functionality, the N6705C can continuously log data to the display and to a file. Data can be simultaneously logged on all four DC outputs. Long battery

Figure 7.1: Keysight N6705C DC Power Analyzer.

life is extremely important for IoT devices: we can perform battery drain analysis using the instrument source measurement unit. Fig-

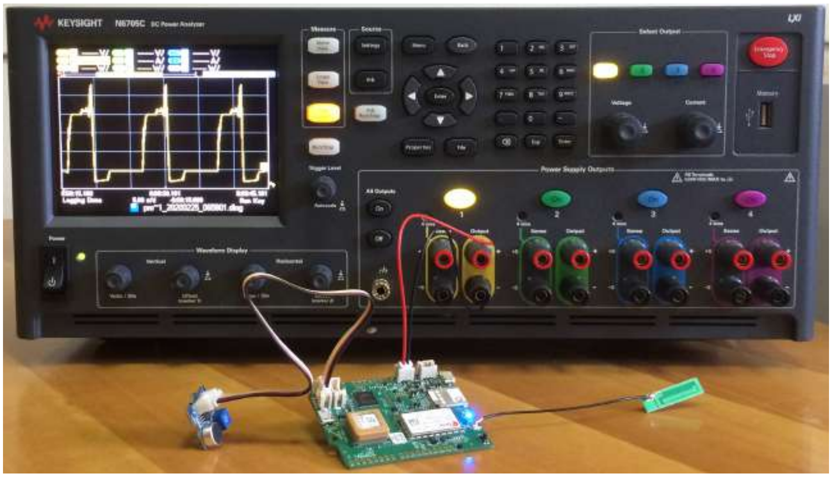


Figure 7.2: Power Consumption Measurement Set Up.

ure 7.2 shows the measurement set up: the board is power by the instrument and using the data log functionality we can measure the current consumption over time.

Measurements of the device power consumption have just begun due to long waiting times to get demo instrument. Figure 7.3 shows the measurement of the switching between low-power mode and working mode utilizing the data log functionality. Figure 7.4 shows a screen capture of the instrument while measuring.

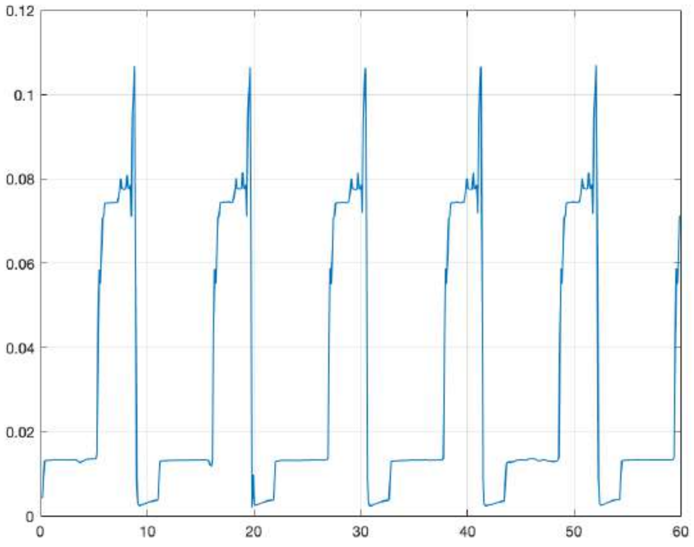


Figure 7.3: Switching Between Low-Power Mode and Working Mode.

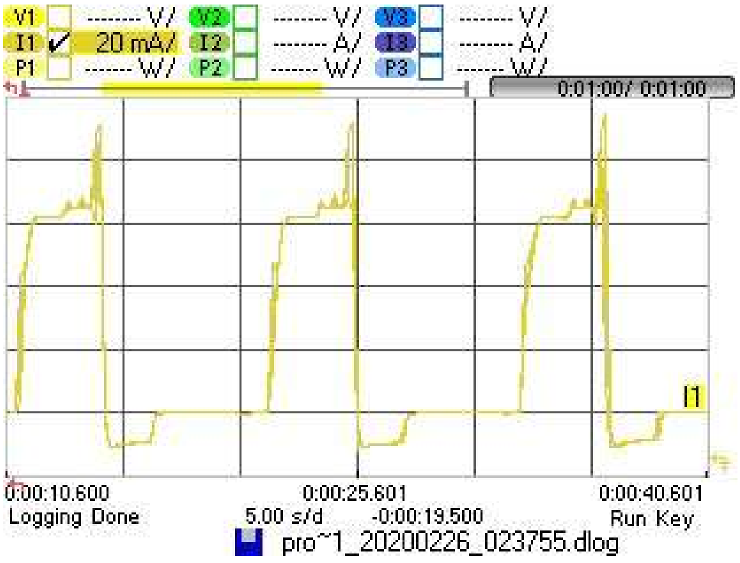


Figure 7.4: N6705C Measurement Scree Capture.

Appendix A

RISC-V and Rocket Core Architecture

This chapter describes the study we carried out about RISC-V. The first section presents an overview of the RISC-V ISA describing the instruction format and the extension mechanism. The second section introduces the Rocket Core processor together with the Rochet Chip SoC generator, presenting some of the main features and describing the design flow.

A.1 Risc-V ISA

RISC-V is an open ISA, that was initially developed by the University of California Berkeley (UCB) and it is now managed by the RISC-V Foundation. Designed following the Reduced Instruction Set Computer (RISC) principles, this ISA started as a project for computer architecture research and education, but is aiming at becoming a standard for industrial implementations [14]. RISC-V is now increasingly drawing the attention of both industry and academia, because it offers the possibility to ease the design of processors from the high costs of compiler support, without requiring to resort to expensive commercial ISAs.

RISC-V base integer ("I") ISA defines four basic instruction formats called R-type, I-type, S-type and U-type [14] that are shown Figure A.1.
These instruction format were defined so that the register fields for both the sources ((rs1) and *rs2*) and the destination (*rd*) are kept in the same position for all the formats in order to simplify the decoding hardware. Since one of the design goal of RISC-V is pro-

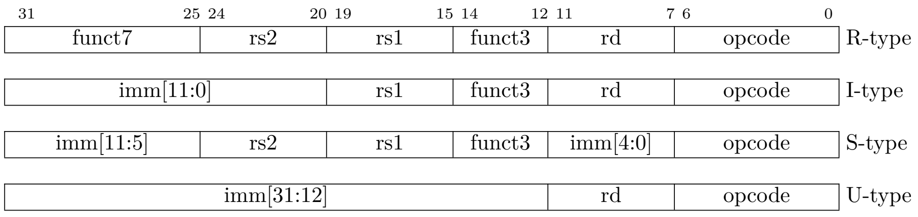
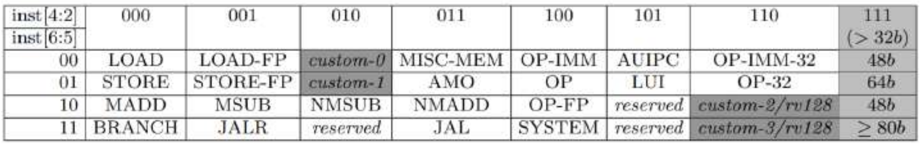


Figure A.1: RISC-V base instruction set.

viding support for extension and customization, the instruction set is organised in a small integer base ("I") plus the optional extension

- "M": standard extension for integer multiplication and division instruction
- "A": standard extension for atomic instruction
- "F": standard extension for single-precision floating point instructions
- "D": standard extension for double-precision floating point instructions
- "C": standard extension for compressed instructions

The ensemble of “IMAFD” extension is indicated with “G” and the resulting ISAs are called RV32G for the 32-bit version and RV64G for the 64-bit one.
RISC-V ISA also enable the presence of non standard extension using four opcodes denoted as *custom-0*, *custom-1*, *custom-2* and *custom-3* that are reserved for custom extension.
custom-0 and *custom-1* are recommended for use by custom instruction-set extensions within the base 32-bit instruction format while *custom-2* and *custom-3* are reserved for 128-bit ISA extension.



inst[4:2] / inst[6:5]	000	001	010	011	100	101	110	111 (> 32*b*)
00	LOAD	LOAD-FP	*custom-0*	MISC-MEM	OP-IMM	AUIPC	OP-IMM-32	48*b*
01	STORE	STORE-FP	*custom-1*	AMO	OP	LUI	OP-32	64*b*
10	MADD	MSUB	NMSUB	NMADD	OP-FP	*reserved*	*custom-2/rv128*	48*b*
11	BRANCH	JALR	*reserved*	JAL	SYSTEM	*reserved*	*custom-3/rv128*	≥ 80*b*

Figure A.2: RISC-V base opcode map. Source [14]

A.2 Rocket Chip SoC Generator and Rocket Core

This section introduces the Rocket Chip SoC generator and the Rocket Core, the main design tools and components around which this work was developed. These two components are linked together, since the Rocket Chip generator is used to configure and generate the Rocket Core.

A.2.1 Rocket Chip SoC Generator

Rocket Chip is an open source SoC generator designed by the Berkeley Architecture Research (BAR) group of the University of California Berkeley (UCB). It is a tool that emits synthesizable RTL, devised to enable designers to build and customise their own SoCs based around the RISC-V ISA [15]. This project is developed in the Chisel language, a dialect of Scala designed for hardware construction and generation. The SoC generator allow the composition of modular designs by using the parametrisation features of the Chisel language. It offers the possibility of tuning and personalising a lot of configuration parameters of the design, including the support for different standard extension of the ISA, in order to generate different cores able to suit the need of the designer. The basic design flow allows the integration of new hardware component described in Chisel, the creation of new custom configurations, the compilation of the Chisel/Scala sources to generate C++ models for cycle accurate simulations and the generation of the synthesizable Verilog RTL sources for pushing the design in the standard industry CAD tools.

A.2.2 Rocket Core

Rocket core is both a processor generator and a library of processor components. As a generator it is able to produce a family of processor core designs, with different configuration parameters. The generated processors have the classical six stage in-order scalar pipeline (Figure A.3) and can implement the base integer 32-bit RISC-V ISA as well as the 64-bit one [15]. The cores have a Memory Manage-

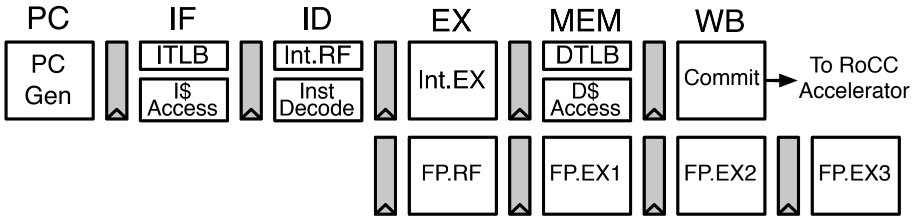


Figure A.3: Pipeline stages. Source [15]

ment Unit (MMU) with optional paged virtual memory support, a configurable non blocking private data cache and a front-end with a configurable branch predictor. The generator exposes numerous parameters for configuring the core, among them there is the possibility to add the support for some optional standard ISA extensions (M, A, F, D) and determining the size of the caches [15]. Figure A.4 shows aa example of a Rocket Chip istance. It features two tiles

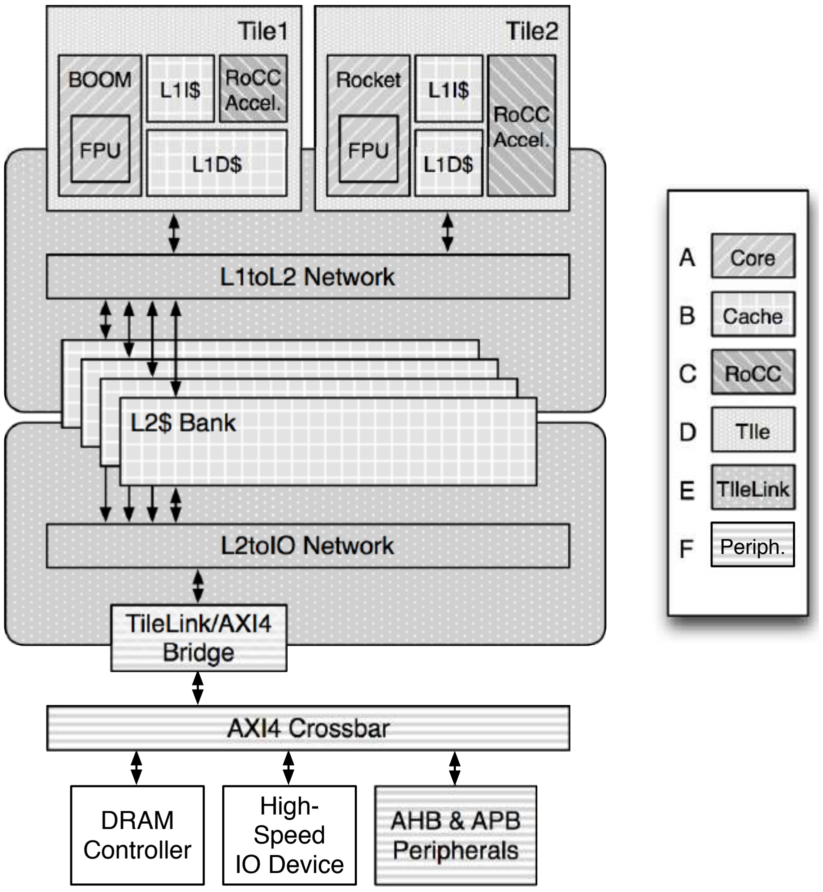


Figure A.4: Example of a Rocket Chip istance. Source [15]

attached to a 4-bank L2 cache that is itself connected to the external I/O and memory systems with an AXI interconnect. Tile 1 is an out-of-order BOOM core with an FPU, L1 instruction and data caches, and an accelerator implementing the RoCC interface. Tile 2 is similar, but it uses a different core, Rocket, and has different L1 data cache parameters.

A.3 Design Flow

In this work we follow the design flow decribed in [16] and [17] shown in Figure A.5 In particular, the path we follow is show in Figure A.6

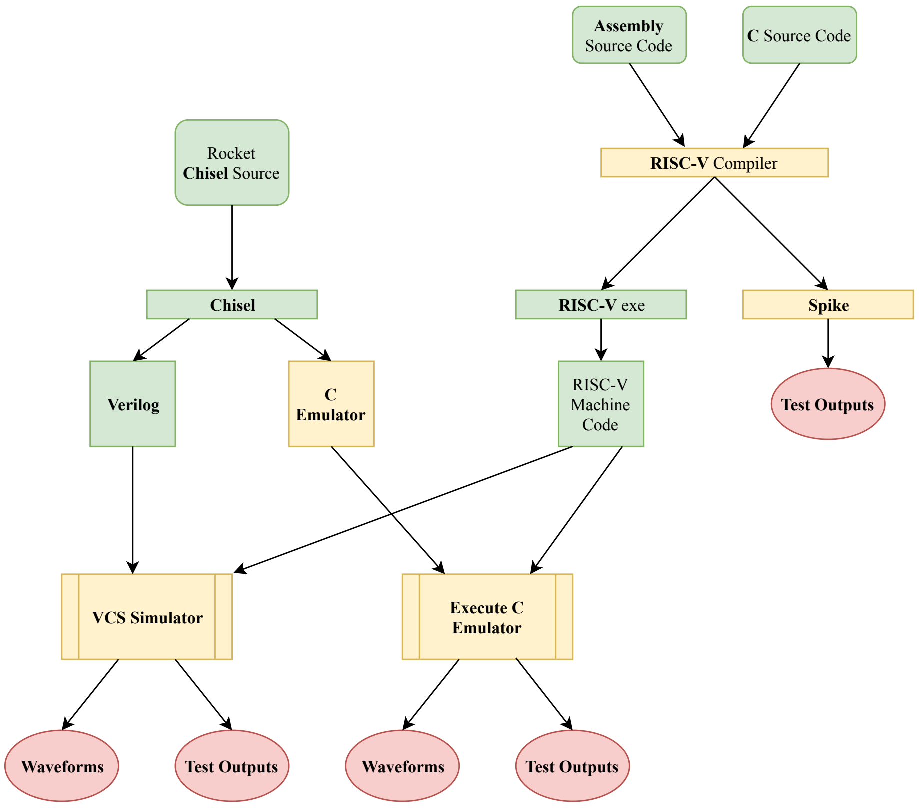


Figure A.5: Rocket Project Flow. Source [16]

We start from a C source code. We can generate executable files according to RISC-V libraries compiling the C source code using one

of the several *ELF* (Executable and Linkable Format) provided by the toolchain. Then we can run an RTL simulation using Synopsys VCS to execute the RISC-V exe on the Rocket chip.
we used the Rocket chip *DefaultConfig* which include one big core, a memory port, an MMIO port, and two external interrupts. The big core includes:

- FPU: a Floating Point Unit
- BTB: a Brench Target Buffer
- MULTDIV: a Multiplication/Division module
- ICache size: an Instruction Cache whit 64 sets, 1 way , 128 row bits, and 4 TLB entries
- DCsche size: a data cache with 64 sets, 4 ways, 64 row bits, and 32 TLB entries

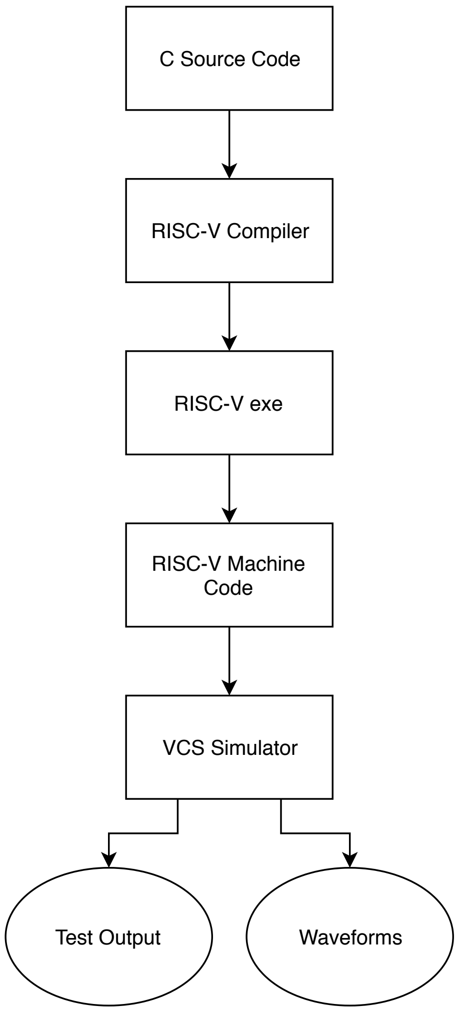


Figure A.6: Design Flow followed in this work.

Appendix B

Software Implementation

This chapter shows the software implementation of the algorithm to acquire geolocalized noise data and send them to the datacentre.

```
#include <Sodaq_UBlox_GPS.h>
#include <Sodaq_R4X.h>
#include <Sodaq_wdt.h>
#include <string.h>

//#if defined(ARDUINO_SODAQ_AUTONOMO)
///* SODAQ AUTONOMO + SODAQ NB-IoT R41XM Bee */
//#define DEBUG_STREAM SerialUSB
//#define MODEM_STREAM Serial1
//#define powerPin BEE_VCC
//#define enablePin BEEDTR
//
//#elif defined(ARDUINO_SODAQ_SARA)
/* SODAQ SARA AFF*/
#define DEBUG_STREAM SerialUSB
#define MODEM_STREAM Serial1
#define powerPin SARA_ENABLE
```

```
#define enablePin SARA_TX_ENABLE

//#elif defined(ARDUINO_SODAQ_SFF)
///* SODAQ SARA SFF*/
//#define DEBUG_STREAM SerialUSB
//#define MODEM_STREAM Serial1
//#define powerPin SARA_ENABLE
//#define enablePin SARA_TX_ENABLE
//
//#else
//#error "Please use one of the listed boards or add your
    board."
//#endif

#define DEBUG_STREAM SerialUSB
#define DEBUG_STREAM_BAUD 115200

#define STARTUP_DELAY 5000

// NL VODAFONE NB-IoT
const char* apn = "nb.inetd.gdsp";
const char* forceOperator = "22210"; // optional -
    depends on SIM / network
const char* urat = "8";

Sodaq_R4X r4x;
//Sodaq_LSM303AGR AccMeter;

static Sodaq_SARA_R4XX_OnOff saraR4xxOnOff;

#include "SOS_A_float.h"
#include "SV_A_float.h"

#define NS 20000

void sendMessageThroughUDP(String value) {

DEBUG_STREAM.println();
DEBUG_STREAM.println("Sending message through UDP");

int localPort = 16666;
int socketID = r4x.socketCreate(localPort);
```

```
DEBUG_STREAM.print("socketID = ");
DEBUG_STREAM.println(socketID);

if (socketID >= 7 || socketID < 0) {
DEBUG_STREAM.println("Failed to create socket");
return;
}

DEBUG_STREAM.println("Created socket!");

String deviceId = "VfbTrH3ejfA78Io9bHBGk5rO";
String token = "vodafone:4
    HLg85WInLyVG0lqFxJlq4FR97OnLLZyazetNd41";
String reading = deviceId + '\n' + token + '\n' + value;

uint8_t size = reading.length();
r4x.socketSend(socketID, "40.68.172.187", 8891, (uint8_t
    *)reading.c_str(), size);
r4x.socketClose(socketID);
}

void setup() {

sodaq_wdt_safe_delay(STARTUP_DELAY);

SerialUSB.begin(115200);
while (!SerialUSB) {}
SerialUSB.println("press enter...");
while (!SerialUSB.available());

sodaq_gps.init(GPS_ENABLE);

initADC();

MODEM_STREAM.begin(r4x.getDefaultBaudrate());

DEBUG_STREAM.println("Initializing and connecting... ");

r4x.setDiag(DEBUG_STREAM);
r4x.init(&saraR4xxOnOff, MODEM_STREAM);
r4x.connect(apn, urat);

if (!r4x.connect(apn, urat)) {
```

```
DEBUG_STREAM.println("FAILED TO CONNECT TO MODEM");
}

initSleep();
}

void loop() {
int i = 0;

float input;

float y1 = 0;
float y2 = 0;
float y3 = 0;
float buf11 = 0.0;
float buf21 = 0.0;
float buf12 = 0.0;
float buf22 = 0.0;
float buf13 = 0.0;
float buf23 = 0.0;

float squarer;

float pr2 = 20e-6 * 20e-6;

float Leq;

float sum = 0;

float lat;
float lon;

char a[13];
String value;


sodaq_wdt_reset();                    // restting the
    watchdog
sodaq_wdt_disable();

digitalWrite(GPS_ENABLE, HIGH);

SerialUSB.println("waiting in loop() ...");
```

```
if (sodaq_gps.scan(true))
{
SerialUSB.print("We are at latitude ");
lat = sodaq_gps.getLat();
lon = sodaq_gps.getLon();
SerialUSB.print(lat, 13);
SerialUSB.print(" longitude ");
SerialUSB.println(lon, 13);
} else {
SerialUSB.println("NO SCAN!");
}
digitalWrite(GPS_ENABLE, LOW);

for (i = 0; i < NS; i++) {
ADC->CTRLA.bit.ENABLE = 0x01;
syncADC();
ADC->SWTRIG.bit.START = 1;
while (ADC->INTFLAG.bit.RESRDY == 0);
syncADC();
ADC->CTRLA.bit.ENABLE = 0;
syncADC();

input = (float)(ADC->RESULT.reg) / 4095;

y1 = SV_A[0] * SOS_A[0][0] * input + buf11;
buf11 = SV_A[0] * SOS_A[0][1] * input - SOS_A[0][4] * y1
    + buf21;
buf21 = SV_A[0] * SOS_A[0][2] * input - SOS_A[0][5] * y1;

y2 = SOS_A[1][0] * y1 + buf12;
buf12 = SOS_A[1][1] * y1 - SOS_A[1][4] * y2 + buf22;
buf22 = SOS_A[1][2] * y1 - SOS_A[1][5] * y2;

y3 = SOS_A[2][0] * y2 + buf13;
buf13 = SOS_A[2][1] * y2 - SOS_A[2][4] * y3 + buf23;
buf23 = SOS_A[2][2] * y2 - SOS_A[2][5] * y3;

squarer = y3 * y3;

sum += squarer;

}
```

```
Leq = 10 * log10((sum / NS) / pr2);
SerialUSB.println(Leq);

memcpy(&a[0], &Leq, sizeof(Leq));
memcpy(&a[4], &lat, sizeof(lat));
memcpy(&a[8], &lon, sizeof(lon));
a[12] = 0;
value = String(a);

sendMessageThroughUDP(value);


y1 = 0.0;
y2 = 0.0;
y3 = 0.0;

buf11 = 0.0;
buf21 = 0.0;
buf12 = 0.0;
buf22 = 0.0;
buf13 = 0.0;
buf23 = 0.0;

sum = 0;

sodaq_wdt_enable(WDT_PERIOD_8X);      // watchdog expires
    in ~8 seconds
systemSleep();
}

static __inline__ void syncADC() __attribute__((
    always_inline, unused));
static void syncADC() {
while (ADC->STATUS.bit.SYNCBUSY == 1);
}

void initADC() {
SYSCTRL->OSC8M.bit.PRESC = 0;                              //
     no prescaler (is 8 on reset)
SYSCTRL->OSC8M.reg |= 1 << SYSCTRL_OSC8M_ENABLE_Pos;     //
     enable source

REG_PORT_DIRCLR1 = PORT_PA11;
```

```
PORT->Group[0].PINCFG[11].bit.PMUXEN = 1;
PORT->Group[0].PMUX[5].reg = PORT_PMUX_PMUXO_B;

GCLK->GENDIV.bit.ID = 0x03;
GCLK->GENDIV.bit.DIV = 9;
GCLK->GENCTRL.reg = GCLK_GENCTRL_GENEN |
    GCLK_GENCTRL_SRC_OSC8M | GCLK_GENCTRL_ID(3); //
    GCLK_GENCTRL_DIVSEL don't need this, it makes divide
     based on power of two
while ( GCLK->STATUS.reg & GCLK_STATUS_SYNCBUSY ) {}

PM->APBCSEL.bit.APBCDIV = 0;                          //
      no prescaler
REG_PM_APBCMASK |= PM_APBCMASK_ADC;

GCLK->CLKCTRL.reg |= GCLK_CLKCTRL_CLKEN |
    GCLK_CLKCTRL_GEN(3) | GCLK_CLKCTRL_ID(30);
while (GCLK->STATUS.bit.SYNCBUSY == 1);

REG_ADC_REFCTRL = ADC_REFCTRL_REFSEL_INTVCC0; //set vref
     for ADC to VCC
syncADC();
// Sampling time, no extra sampling half clock-cycles
REG_ADC_SAMPCTRL = ADC_SAMPCTRL_SAMPLEN(0);
//
// Input control and input scan
REG_ADC_INPUTCTRL |= ADC_INPUTCTRL_GAIN_1X |
ADC_INPUTCTRL_MUXNEG_GND |
ADC_INPUTCTRL_MUXPOS_PIN19;

ADC->CTRLB.bit.RESSEL = 0;
syncADC();
ADC->CTRLB.bit.PRESCALER = 0x0; //DIV4
syncADC();
ADC->CTRLA.bit.ENABLE = 0x01;
syncADC();
ADC->CTRLA.bit.ENABLE = 0;
syncADC();
}

void initSleep()
{
// Set the sleep mode
```

```
SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk;
}

/**
Powers down all devices and puts the system to deep sleep
    .
*/
void systemSleep()
{

__WFI(); // SAMD sleep

}
}
```

Bibliography

[1] EU Directive. Directive 2002/49/ec of the european parliament and the council of 25 june 2002 relating to the assessment and management of environmental noise. *Official Journal of the European Communities, L*, 189(18.07):2002, 2002.

[2] Birgitta Berglund, Thomas Lindvall, Dietrich H Schwela, World Health Organization, et al. Guidelines for community noise. 1999.

[3] Charlotte Hurtley. *Night noise guidelines for Europe.* WHO Regional Office Europe, 2009.

[4] EC WHO, World Health Organization, et al. Burden of disease from environmental noise. *Burden of disease from environmental noise*, 2:255–270, 2011.

[5] EU Commission et al. The green paper on future noise policy. *COM (96)*, 540, 1996.

[6] European Commission Working Group et al. Assessment of exposure to noise. good practice guide for strategic noise mapping and the production of associated data on noise exposure, 2007.

[7] Nicolas Maisonneuve, Matthias Stevens, Maria E Niessen, and Luc Steels. Noisetube: Measuring and mapping noise pollution with mobile phones. In *Information technologies in environmental engineering*, pages 215–228. Springer, 2009.

[8]

[9] Enda Murphy and Eoin A King. Testing the accuracy of smartphones and sound level meter applications for measuring environmental noise. *Applied Acoustics*, 106:16–22, 2016.

[10] Xavier Sevillano, Joan Claudi Socoró, Francesc Alías, Patrizia Bellucci, Laura Peruzzi, Simone Radaelli, Paola Coppi, Luca Nencini, Andrea Cerniglia, Alessandro Bisceglie, et al. Dynamap–development of low cost sensors networks for real time noise mapping. *Noise Mapping*, 3(1), 2016.

[11] *International Electrotechnical Commission. Electroacoustics - Sound level meters - Part 1: Specifications. IEC 61672-1:2013.*

[12] Cyril M Harris. *Handbook of acoustical measurements and noise control.* McGraw-Hill New York, 1991.

[13] Steve Winder. *Analog and digital filter design.* Elsevier, 2002.

[14] Andrew Waterman, Yunsup Lee, DA Patterson, and Krste Asanovic. The risc-v instruction set manual, volume i: User-level isa, version 2.2, eecs department. *University of California, Berkeley*, 2017.

[15] Krste Asanovic, Rimas Avizienis, Jonathan Bachrach, Scott Beamer, David Biancolin, Christopher Celio, Henry Cook, Daniel Dabbelt, John Hauser, Adam Izraelevitz, et al. The rocket chip generator. *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17*, 2016.

[16] *V. Ciotoli, "Design of a RISC-V Based Accelerator For Image Pocessing", Master Thesis. 2019.*

[17] *L. Calicchia, "Approximated Floating-Point Accelerator for Energy Efficient Embedded Systems", Master Thesis. 2019.*