System for Accurate Acoustic Presentation of Noise Sources

Authors:
Esben MADSEN
Jorge COFIÑO CÓRCOLES

Supervisor:
Søren KRARUP OLESEN

AALBORG UNIVERSITET

Aalborg University
Acoustics
Department of Electronic Systems
Frederik Bajers Vej 7
9220 Aalborg Ø
Telephone 96 35 86 00
http://acoustics.es.aau.dk/

Title: System for Accurate Acoustic Presentation of Noise Sources

Project term:
Master project, spring semester 2010

Project group:
10gr1063

Group members:
Esben Madsen
Jorge Cofiño Córcoles

Supervisor:
Søren Krarup Olesen

Copies: 5

Pages: 145

Appendices: 1 CD with software and sources.

Finished 3/6 2010

Synopsis:

In this thesis, a system that is able to reproduce the sound from noise sources as well as show the different levels present at different distances and wind speeds has been designed.
This noise presentation system is composed by a small computer (nanoLIAB) and a user interface designed for it. An overview of the current Danish legislations regarding noise evaluation of several kinds is given so it is known what the guidelines to measure wind turbine noise are.
Next, an user guide on how to carry out measurements of wind turbine noise according to the standard IEC 61400 is presented and a measurement following this guide has taken place. The recorded data is evaluated to find possible artifacts that may classify the signal as "corrupt" or "clean" and a calibration of the system is done. The designed noise presentation system plays back the sound recorded from a wind turbine in combination with a set of parameters that characterize the reproduced signal. The user interface provides a good overview of what is being played, by showing the user relevant data about the source that is being played and enables the user to change the settings and thereby playing a different sound.

This report is made by:

Esben Madsen

Jorge Cofiño Córcoles

Contents

Introduction

Project Description

In the modern society, noise from a number of different sources such as roads, railways, airports ad wind turbines, among others, gives rise to annoyance in the neighboring communities. The different noise sources that surround us oblige to undertake an action about how o regulate them The thesis concerns the noise produced by wind turbines and its evaluation and limit assessments.

Wind turbines generate noise from multiple mechanical and aerodynamic sources. With the evolution of the design and construction of wind turbines, these have become much quieter. Wind turbines are often located in rural or remote areas where low noise levels are present. Furthermore, while noise may be a concern to the public living near wind turbines, much of the noise emitted from the turbines is masked by background noise. However, the noise emitted from wind turbines is still a public concern. The problems associated with wind turbine noise have been identified regarding annoyance on resident areas and several actions are proposed when performing an environmental impact assessment. Noise levels can be measured but, as it happens with other noisy sources such as airports, the public's perception of the noise impact of wind turbines is mainly a subjective determination.

Noise is defined as any unwanted sound and depends on several factors like the following [Org95, Chapter 2],

- Level of intensity, frequency, frequency distribution and patterns of the noise source
- Background noise levels
- Distance between the emitter and receptor
- The nature of the noise receptor (resident areas, working places, groups of animals. . .)

The effects of noise on people can be classified into three general categories [Org95, Chapters 3 and 4]:

- Subjective effects including annoyance, nuisance, dissatisfaction
- Interference with activities such as speech, sleep, and learning
- Physiological effects such as anxiety, tinnitus, or hearing loss

The sound levels associated with wind turbines produce effects only in the first two categories. Workers in industrial plants, and those who work around aircraft can experience noise effects in the third category. Whether a noise is objectionable will depend on the type of noise (tonal, broadband, low frequency, or impulsive) and the circumstances and sensitivity of the person (or

receptor) who hears it. Because of the wide variation in the levels of individual tolerance for noise, there is no completely satisfactory way to measure the subjective effects of noise or of the corresponding reactions of annoyance and dissatisfaction.

In order to aid in the understanding of what a certain noise level means, it is of interest to obtain a system which can accurately present this by visual and auditory means. Such system could be helpful to public decision makers when assesing the impact of a given noise source.

This thesis reviews how noise levels are presented prior to a legal evaluation. A reliable assessment and presentation of the noise levels as a consequence of the activity produced in different facilities (wind turbines, roads, airports. . .) is of vital importance when evaluating if a new facility is suitable of being implemented or remodeled.

The problem of noise affects every social layer within a community in different ways (business opportunities, health, preservation of scholar areas. . .). The political organizations have the final decision on enforcing the limits but are not familiar with what these limits mean. Usually, noise levels are presented simply by a single number, a value in decibels (dB). These dB values are often weighted as dB(A), a correction to adapt the noise level according to human hearing characteristics. The decision is made upon this number and nothing else.

This thesis has been focused on the study of noise permittivity, proposing a new method which satisfies the current demands within this field. Thus, arises the necessity of implementing an unique, single, portable and cheap device capable of reproducing these noise levels in a reliable way, adding features such as visuals which may enhance the perception of the noise produced by a given source. The techniques used can be applied for any kind of noise source (roads, airports, trains. . .) although the research done is principally focused on the noise generated by wind turbines.

Problem Statement

> The goal of this thesis is to design a device capable of presenting a noise source accurately by auditory and visual means. The scope will be limited to wind turbine noise.

Preliminary Decisions

- A revision of noise standards and local legislation will be carried out. The most interesting aspects are those related with wind turbine noise. Nonetheless, other noise sources such as roads, railways or airports turn out to be interesting as well. So, a revision of the corresponding standards and local legislations will be also presented for these noise sources. Although this thesis will be centered on noise from wind turbines, it is intended to show that what is done for wind turbines, can also be done for other sources.

- Wind turbine noise measurements can be carried out following the standard IEC61400 [Com09]. According to the Danish legislation on this matter, Announcement 1518 [Mil06a] defines a measurement procedure according to the mentioned standard. However, not all guidelines are considered (e.g. measurement positions). This thesis is intended to fit on the directions given by the current Danish legislation so that the final product suits as much as

possible with the current local necessities. For this, and for simplicity reasons also, it has been decided to follow the Announcement 1518.

- Measurements of the noise will be carried out at a wind farm located close to Aalborg. The wind turbines are land based, far enough from any other disturbing noise sources such as roads. Prior to the measurements, a permission will be requested to the owner of the property.

Part I

Initial Analysis

In the initial analysis of the problem, the project focuses on two main points:

- What are the relevant fields of the project?
- Which requirements does that give to the final system?

When looking at the relevant fields, the initial examination will try to give a short overview of the laws and standards relevant to the single fields. Relevant information about the public decision making on the field and how noise is currently presented to the decision makers will then be given.

Once an overview of the fields has been established, the basic requirements to an implemented system that can be of aid in the decision making will be defined. This includes which measures are relevant to present and which the costumer should be able to manipulate.

1 Relevant fields

In this chapter, some fields where the project could be of interest are examined.

The following fields have been chosen for further examination:

- Road noise
- Railways noise
- Airport noise
- Wind turbines

All of these fields share the fact that when establishing a new regulation, noise is a relevant part of the decision making because neighbors are affected by it.

Currently, the noise is evaluated by professionals in environmental assessments, where measures of noise are calculated in dB and curves of the limits given by laws drawn on a map.

In this chapter, an overview of the relevant laws and standards of the fields will be given and more details of the actual decision making will be provided

Finally, a short summary of the fields with discussions of their differences and similarities will be provided. This will along with a limitation to a single field give the basis for the further direction of the project and provide the relevant information for setting basic requirements of the final implementation.

Internationally the standard ISO 1996-1:2003 [fS03] is used as a general guideline for noise and measurements

1.1 General situation in noise assessment in Denmark

In 2007-09, a number of municipalities and other authorities have carried out a noise mapping, which culminated in the so-called noise maps. The mapping is done using guidelines in the noise

notice which implements the EU noise directive from 2002 [otEC02b]. The next phase of the noise mapping in 2012 will include significantly more roads and railways, and beyond the Copenhagen area are also Odense, Aalborg and Aarhus mapped in detail.

The assessment of noise limits is carried out in relation to the Environmental Protection Agency's recommended limits [Mil07a]. There are no similar limits for different types of noise, because the different noise types are not experienced equally annoying. Railway noise is not as annoying as road noise and aircraft noise experienced as most annoying.

The indicative levels for noise in residential areas are:

- Road traffic noise: 58 dB L_{den}
- Railway noise: 64 dB L_{den}
- Aircraft noise: 55 dB L_{den}
- Wind turbine noise: 44 dB L_{Amax}

Noise from businesses must be assessed in relation to the specific guidance values for companies that apply separately for day, evening and night time, and is also tougher on weekends than on weekdays. For clarity sake, the Environmental Protection Agency indicated levels of planning use for businesses that like the other limits are expressed in days value.

The limit for planning the use of floor areas for residential and mixed residential and commercial are:

- Noise from businesses: 50 dB L_{den}

There is no guidance values for night noise, L_{night}. The Environmental Protection Agency's Guide 4, 2006 [Mil06b] on noise mapping and noise action plans it is stated that, based on the available knowledge can be expected 15% risk of sleep disorders at levels of:

- Road traffic noise: 52 dB L_{night}
- Railway noise: 62 dB L_{night}
- Aircraft noise: 53 dB L_{night}
- Wind turbine noise: 44 dB L_{Amax}

1.2 Road Noise

Directive 70/157/EEC regulates the noise emitted by any vehicle intended for use on the road. It has been review several times up to its latest version, 2007/34/EC [otEC07], issued on June 14^{th} 2007.

This Directive and its predecessors apply specifically to any motorized vehicle with at least four wheels, and a maximum speed of at least 25 km/h. Regarding the noise level of the mechanical parts and other systems of the vehicles concerned, the limits are shown on *Table 1.1*.

Vehicle categories	Noise limit (dB(A))
Vehicles intended for the carriage of passengers, and comprising not more than nine seats including the driver's seat	74
Vehicles intended for the carriage of passengers and equipped with more than nine seats, including the driver's seat; and having a maximum permissible mass of more than 3,5 tonnes and:	
with an engine power of less than 150 kW	78
with an engine power of not less than 150 kW	80
Vehicles intended for the carriage of passengers and equipped with more than nine seats, including the driver's seat; vehicles intended for the carriage of goods:	
with a maximum permissible mass not exceeding 2 tonnes	76
with a maximum permissible mass exceeding 2 tonnes but not exceeding 3.5 tonnes	77
Vehicles intended for the carriage of goods and having a maximum permissible mass exceeding 3,5 tonnes:	
with an engine power of less than 75 kW	77
with an engine power of not less than 75 kW but less than 150 kW	78
with an engine power of not less than 150 kW	80

Table 1.1: Noise limits assessed for road traffic by the European Union

1.2.1 Noise Measurement in Roads

There are several methods to measure the noise emitted by vehicles in Europe. These methods vary depending on the different Member States, e.g. given by the the European Acoustics Association (EAA) [Ass02] are the CRTN (United Kingdom), Harmonoise/Imagine (EU), RLS90/VBUS (Germany) or Nord 2000 (Scandinavia). Methods Nord 2000 and Harmonoise/Imagine are considered by the EAA the most suitable ones regarding the performance of all or the majority of the criteria to be taken into account for the evaluation of the noise emitted by motorized vehicles [KP09].

Nord 2000

Nord2000 is the method followed in Denmark and in all Scandinavia for the assessment of noise from road and railways traffic. Nord 2000 describes the source strength as the sound power level as a function of speed for different vehicle categories [KJPS06]. Frequency band data are available for the range 25-10000 Hz. The sound power levels have been determined from measurements of the SEL-level during pass by and the source model has been used to calculate the equivalent omni directional sound power level. For different vehicle categories the energy average of many pass-bys has been determined in steps of 5 km/h.

1.2.2 Relevant Legislations

The Guide No. 4, 2007 [Mil07b] from the Danish Environmental Protection Agency concerns the calculation of noise exposure levels and the establishment of adequate levels to preserve a suitable acoustic environment with respect to the noise generated by road traffic. The Danish Environmental Protection Agency has set guidance values for road traffic noise on different types of areas. The indicative limits are formulated by the indicator L_{den}, used for noise mapping and planning, and apply the annual mean value of outdoor noise in free field [Mil07b, Chapter 2, p. 13]. L_{den} is an indicator that attaches noise events in the evening and night time more weight than the noise on days, and it has a better consistency with the way the noise experienced at than average noise, L_{Aeq} [KJPS06, Chapter 2, pp.10-11]. Therefore, the noise limits are not directly comparable with the previous limit, which is expressed by L_{Aeq}.

The guidance values for road traffic noise are shown on *Table 1.2*

Area	Noise limit
Recreational areas in the open country, cottage areas, campsites and the like	L_{den} 53 dB
Residential areas, kindergartens, nurseries, schools and teaching buildings, nursing homes, hospitals etc. Besides allotments, outdoor lounge areas and parks.	L_{den} 58 dB
Hotels, offices, etc...	L_{den} 63 dB

Table 1.2: Noise limits assessed for road traffic

Situation regarding housing planning

Several situations are considered regarding the mitigation of noise for the next areas which a municipality may have [Mil07b, Chapter 3, pp. 17-19].

Planning for new houses and similar noise sensitive uses. In this situation it is important to emphasize the need to schedule an attractive noise environment where the noise is lower than 58 dB. This way, it is remarked that municipalities offer quiet areas, far from major roads and other noise sources providing an attractive urban environment where the noise is lower than the indicative limits. Also, the inclusion of parks and recreational areas is convenient in the design of these quiet areas contributing to the experience of an attractive environment.

New areas with noise. New, dense urban areas and near the roads, e.g. former industrial and port areas are transformed into new residential areas to mixed urban functions, are noise affected areas which were not originally designed to noise-sensitive uses.

The action to be taken on these new areas requires a detailed noise map of the area to investigate to what extent noise removal from the road can be reduced (interference at source). It can be the case when there are both structural and traffic changes same time. To ensure a noise level below 58 dB, noise barriers and other measures shall be placed throughout the area.

If there are new dense urban areas, the area's own traffic on the internal road network will be so great that the noise level exceeds the indicative limit, unless special measures are taken. In this situation, planning considerations include with to improvement of the urban environment by establishing for instance quiet zones such as parks or recreational zones.

New homes in existing noise congested urban areas. This aspect includes urban renewal and refreshing of existing residential areas in cities and hole filling in a housing block. It may be old, central neighborhoods, where added feature of new homes or renewal of existing residential areas close to busy roads. In these areas there is great noise, and there is an increased risk of negative health effects.

To try to approximate the levels closer to the demanded 58 dB, in urban areas can be useful to apply noise-reducing road surfaces, the possibility to regulate traffic in order to reduce noise impacts on dwellings and the building design with special noise insulation or shielding, so as to ensure an adequate noise indoors even with open windows under 58 dB. However, residences should not be under any circumstances planned when noise levels are higher than 68 dB.

Situation regarding new roads, extensions and traffic changes

The Danish Environmental Protection Agency assess planning of new roads and road upgrades the same way as when planning for new housing. Besides, should take into account the noise conditions when planned major traffic shifts. There is no reference levels for noise from new roads, but the noise limits specified in *Section 1.2* can be used as assessment criteria [Mil07b, Chapter 3, pp. 20-21].

New roads and road extensions. When establishing new roads, extension or major changes within existing roads, the noise-related implications for the future affected homes should be carefully considered. The new road should be designed to minimize the annoyance caused by the noise using noise-canceling pavements. Also, public aid for noise insulation of the noise affected residences is an important issue that the road authorities may consider. According to the Road Rule Committee [Vej02] it is recommended to ensure that the noise is lower than $L_{Aeq} = 55dB$ in second homes and $L_{Aeq} = 50dB$ at homes when planning and designing new roads. These levels correspond to a day/night noise level, L_{den} of 58 dB and 53 dB respectively.

Besides, when modifying or extending a given roads, a noise impact assessment must be included on the planning of the works so that to comply with the Environmental Protection Agency's recommended limits. It is essential when planning a road expansion to take into account the noise, so that the need for any interference control measure has been clarified as a part of the planning process. There should be noticed that a considerable growth on the traffic may be expected in the future as a consequence of the road-works.

Traffic changes. When planning changes in the traffic flow, change the proportion of heavy traffic or redistribute the total amount of traffic on roads, the noise implications should also be assessed. These traffic changes normally require an agreement between the road authorities and police. All changes affecting traffic flow require the road authorities to identify the consequences for both individuals who are exposed to more noise and for those who suffer less, so the total change can be made up. In situations where conditions deteriorate, it is important the use of instruments to reduce the noise for the affected people like for instance, noise-reducing coatings.

1.3 Railways Noise

For the regulation of the noise emitted by track vehicles the Commission of the European Communities refers to Directive 2001/16/EC [otEC01]. This Directive is not specifically concerned with

regard to noise assessment but covers, among others, the following aspects,

- Infrastructure
- Control, Command and Signalling
- Maintenance
- Traffic operation and Management
- Noise problems

In 2004, the latest review was issued in which high speed trains were included.

1.3.1 Noise Measurement in Railways

Amongst the different methods to measure the noise emitted by these kind of vehicles, it can be remarked Nord2000 (*see Section 1.2.1*), Harmonoise/Imagine and the Standard ISO3095. According to the Danish Legislation, Nord2000 is the method that shall be followed for the assessment of noise in railways (see *Section 1.2.1*).

ISO 3095

The Standard ISO 3095:2005 specifies the conditions needed to carry out measurements of levels and spectra of noise emitted by all kinds of vehicles operating on rails or other types of fixed tracks [Org05]. For frequency analysis of the data, the Standard refers to ISO 266. The results obtained from measurements can be used for,

- Characterizing the noise emitted by trains
- Comparing the noise emission from different vehicles on a certain track section
- Collecting basic source data for the trains

This Standard gives directions for measuring the noise on different circumstances such as,

- Trains moving at constant speed, measuring the A-weighted equivalent continuous pressure level on the pass-by time, L_{pAeq,T_p} for whole trains and parts of them
- Stationary trains, measuring the A-weighted equivalent continuous sound pressure level, $L_{pAeq,T}$

1.3.2 Relevant Legislations

The Guide No. 1, 1997 [Mil97] from the Danish Environmental Protection Agency concerns the calculation of noise exposure levels and the establishment of adequate levels to preserve a suitable acoustic environment with respect to the noise generated by railways traffic. Also, the guidance values for rail noise (see *Table 1.3*) are primarily intended for planning purposes. They are like levels of road noise now expressed by the indicator L_{DEN}.

Area	Noise limit
Recreational areas in the open country, cottage areas, campsites and the like	L_{DEN} 59 dB
Recreation areas in or near urban areas (city parks, allotments, gardens, utility, tourist campsites)	L_{den} 64 dB
Residential areas (residential, day care etc., outdoor lounge areas)	L_{den} 64 dB
Public purposes (hospitals, schools)	L_{DEN} 64 dB
Professional services expenses etc. (hotels, offices)	L_{DEN} 69 dB

Table 1.3: Noise limits settled for railways traffic

There are also requirements for both the maximum level of noise and vibrations of the individual dwellings. The guideline limit for the maximum level is 85 dB (L_{Amax}), and limit vibrations is 75 dB KB-weighted acceleration level. To avoid vibrations, the guide provides a planning-related minimum distance between track center and those land uses (excluding utility gardens). If it can be shown that vibrations can be observed when building close to the track.

1.4 Airport Noise

At an European level, Directive 2002/30/EC [otEC02a] regulates the sound level emitted by aircrafts and generated in airports in order to improve the acoustical environment of the surrounding areas. Thus, it is focused on the sustainable development of the air transport by reducing noise that may affect the different areas located in the vicinity of an airport.

This Directive establishes common rules to prohibit the traffic and promote a gradual withdrawal of the noisiest aircrafts from European airports being a revision of the former one, Directive 925/1999/EC which was intended to solve the problem of noise by placing noise-reducing devices on those noisiest aircrafts. These devices are known as "Hushkits". The main reason of the existence of this new Directive is the loophole that implied the former, which only referred to the "Hushkits" without any mentioning of a hypothetical withdrawal of those aircrafts which use them. Whit this new regulation, all Member States' authorities can prohibit or limit the use of aircrafts whose noise emission does not completely fulfill the requirements according to the regulations given by the International Civil Aviation Organization (ICAO) noise standards. Besides, the airport authorities must provide faithful proof of the acoustical pollution present by carrying out an impact assessment and demonstrate that all possible measures to reduce the noise have been performed.

1.4.1 Noise Measurement in Airports

According to the actual regulation, the measurement of the noise produce at an airport environment is given by the Standard ISO 3891 [fS79], which provides the necessary steps for the acquisition, processing, normalization and reporting of the measured data in two different levels,

1. Data which require an spectral analysis in function of the time
2. Data which only require frequency weighting

Besides, this Standard provides a method to determine the noise exposition during a given time interval.

1.4.2 Relevant Legislations

The Announcement 60274 [Mil94b] from the Danish Environmental Protection Agency concerns the calculation of noise exposure levels and the establishment of adequate levels to preserve a suitable acoustic environment according to noise generated by air traffic.

The calculation method of noise levels is called Minitest and is included on the Announcement. This method is referenced to the Nordic Guidelines for Air Traffic Noise Calculations [sgm08] and includes calculations of equivalent levels establishing different noise contours obtaining the data of groups of two aircrafts on different points (see *Figure 1.1*). The metrics calculated to assess the noise exposure are L_{den} and $L_{max,night(22:00-07:00)}$ for departures, arrivals and taxiing. L_{den} is an indicator of the overall noise level during the day, evening and night which is used to describe the annoyance caused by exposure to noise [otEC02b]. Calculations are made separately and consists of the measuring of two landing and departure maneuvers from 12 different combinations of aircraft types and departure and landing weights and paths.

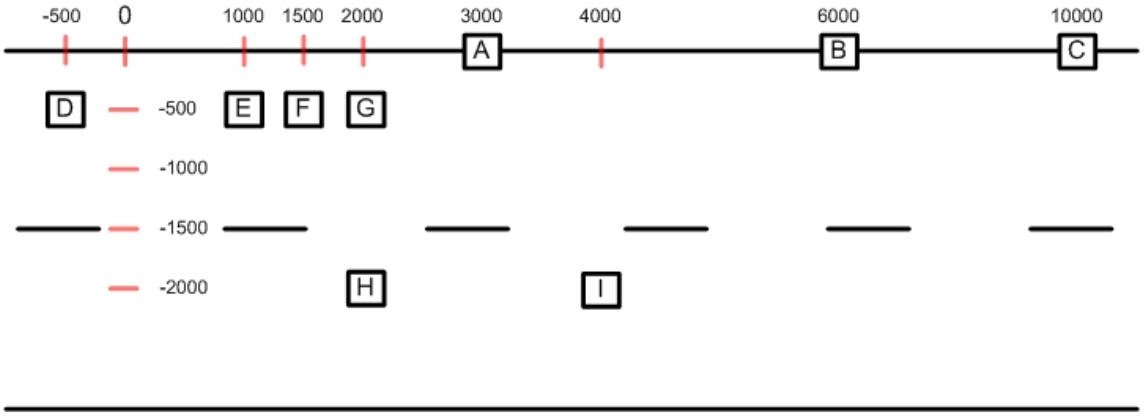

Figure 1.1: Sketch of the measurements to be taken in the runway according to Announcement 60274 from the Danish Environmental Protection Agency. The letters identify the measuring point in the tables of limits listed below

Table 1.4 indicates the threshold values of noise from aircrafts during take-off and landing, taxi driving and from positions on civil airports, public airfields and military air bases [Mil94a, Chapter 4, pp. 41-45].

In residential areas into recreational areas for the night is the maximum value of particular significance. The maximum value of the A-weighted sound pressure level should, for takeoffs and landings at night (22:00-07:00), not exceed 70 dB(A) for public airfields and 80 dB(A) for airports and airfields.

Type of facility	General airfield$_{1)}$	Airport$_{2)}$, Air Base$_{3)}$
residential areas and noise sensitive buildings for public purposes (schools, hospitals, nursing homes, etc.)	45 dB$_{4)}$	55 dB
scattered houses in the open countryside	50 dB	60 dB$_{5)}$
professional services (hotels, offices and similar)	60 dB	60 dB
Recreation areas with accommodation (cottages, allotments, campsites, etc.)	45 dB	50 dB
Other recreational facilities without accommodations	50 dB	55 dB

Table 1.4: Indicative limit values for exposure to noise from outside, take off and land planes, calculated by L_{den} method.
1) "General Airfield" is used for places where traffic is almost entirely composed of general aviation and special classes of traffic as school operations with light aircrafts, parachute flights, ultra-light flight etc.
2) "Airport" is used on sites where a substantial part of the traffic consists of commercial transport of passengers, cargo and mail with large aircrafts.
3) "Air Station" is used on sites that exclusively or primarily operated by the defense aircraft.
4) The limit value of 50 dB is indicative and a general assessment in the specific region is recommended.
5) New homes should not be taken as a starting point where the noise exposure level (L_{den}) exceeds 55 dB or the maximum value (L_{Amax}) is greater than 70dB

1.5 Wind Turbines

When looking at wind turbines, the standard IEC 61400 [Com09] applies to a number of aspects about "Wind turbine generator systems", including safety, design, power performance, noise and load measurements, certification and testing. For this project, IEC 61400-11 Noise Measurement will be most relevant.

The Danish laws on the area is given in Announcement no. 1518 from the Danish Ministry of the Environment [Mil06a]. Appendix 1 of the Announcement concerns the measurements of the noise and states that measurements carried out according to IEC 61400 fulfill the requirements and may be used for calculating the apparent sound power level, $L_{\text{WA,ref}}$.

1.5.1 Noise Measurement

The basic idea of the measurements is as follows, where h is the height of the turbine and d is the diameter of the blades:

1. Record the noise of the wind turbine
2. Normalize to a wind speed in the reference height of 10 m
3. Correct for background noise to achieve $L_{\text{A,ref,k}}$
4. Determine the sound power level of the turbine, $L_{\text{WA,ref}}$
5. Determine the sound pressure level at a given position, L_{pA}
6. Determine audible tonality and noise impact, L_r

The full description of the above with all relevant details and formulas can be found in appendix A. For now it is sufficient to know that when the sound power level of the wind turbine, $L_{\text{WA,ref}}$, is known, the sound pressure level at a given distance in 1.5 m height, L_{pA}, can be calculated.

1.5.2 Relevant Legislations

The Announcement about noise from wind turbines [Mil06a] concerns establishment, modification and commission of wind turbines. The process of approving a new wind turbine or group of turbines a report must, according to the wind turbine announcement [Mil06a], be submitted to the municipal council containing documentation of how the wind turbines will adhere to the noise limits.

The documentation must include

1. A report of measurements of the noise emission from one or more samples of the turbine type
2. Maps of the area including the position of the wind turbines, existing turbines, neighboring residence and distances to these and to other noise sensitive areas
3. Calculations of the noise emission at the above positions

The following is a translation from the wind turbine Announcement, concerning the noise limits, and *Figure 1.2* illustrates the area in which the limits apply:

> The owner of the turbine(s) is responsible for complying with the noise limits:
>
> 1. The most noisy point at outdoor residence areas less than 15 m from neighboring residential[1] must not exceed
>   (a) 44 dB (A) at a wind speed of 8 m/s
>   (b) 42 dB (A) at a wind speed of 6 m/s
> 2. The most noisy point at outdoor residence areas with noise sensitive uses[2] must not exceed
>   (a) 39 dB (A) at a wind speed of 8 m/s
>   (b) 37 dB (A) at a wind speed of 6 m/s

All of the noise levels are determined in accordance with appendix 1 of the wind turbine Announcement [Mil06a] as "equivalent, corrected, A-weighted noise levels in 1.5 meters height". The wind speeds are measured in 10 m height or determined by measuring the output power of the wind turbines and found from known power curves of the turbines. The wind speeds are then corrected to 8 or 6 m/s according to given corrections.

An example of the documentation presented to the councils is a noise curve as seen on *Figure 1.3*. The figure shows a map with the wind turbine positions, neighbor residence and a dotted line, called the noise curve. The curve is a calculated "border" showing the worst case noise, meaning that outside this curve the emitted noise will always be below 45 dB(A) – even in worst case scenarios

[1] Nearby Residential: Any other residential than the wind turbine owner's private living quarters.

[2] Noise Sensitive Use: Areas used for or in local plan or town planning regulations are designated for residential, institution, holiday or allotment garden purposes or as recreation areas.

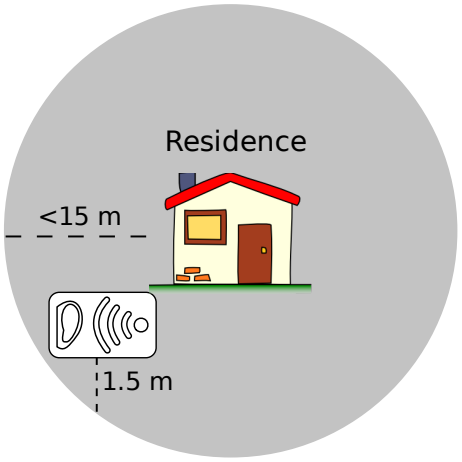

Figure 1.2: Sketch of the area around a residence in which the noise limits from the wind turbine announcement [Mil06a] applies.

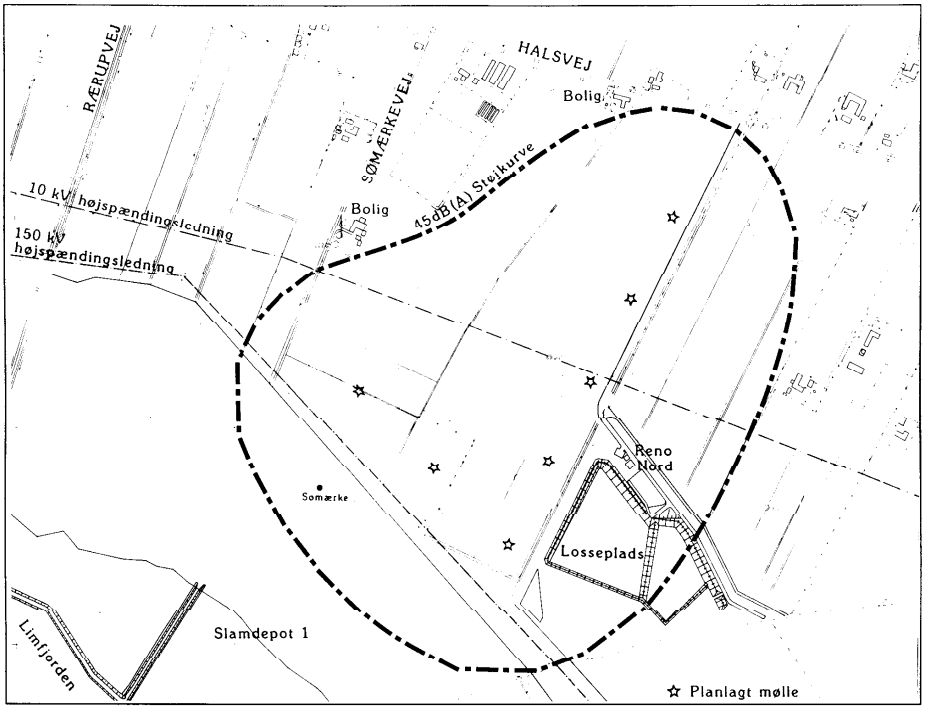

Figure 1.3: 45 dB(A) noise impact curve for a fully built wind farm (calculated) [Kom94]

1.6 Discussion and project limitation

Throughout this chapter, four different noise sources have been covered with regard to which laws and standards should be followed when considering a new facility, and the specific noise limits given by the Danish legislation.

When it comes to the actual decisions, in the cases of new highways, railways and airports, the initial decisions are taken by either the state or regions. For smaller roads, new houses near noise sources or wind turbines, it is always the local administrations who have the authority to allow a new facility. Regardless the type of facility planned, the commune always has the last word regarding the noise since it is responsible for ensuring that the noise limits given by laws and announcements are fulfilled [Mil07b, sec 5.2]. From this, it can be concluded that the main target group of this project are probably the councils of the individual communes.

The limits of road, railway and airport noise are all given as the "A-weighted long-term average day-evening-night noise level", L_{den}, in dB. L_{den} is a single number rating for a noise, which takes the time of day that noise is emitted into consideration, by adding a penalty to noise emitted in the evening and at night. Whereas this rating makes a lot of sense as a single number rating for describing the overall sound impact, it is hard to use this level for predicting the noise level at a given time, without knowing a lot of extra variables, such as the amount and type of cars, trains or airplanes and their distribution throughout the day.

When looking at the wind turbine legislation, the noise limits given are "all time maximum" levels and it will therefore be easier to predict the specific levels of a sound intended for presenting the level to a person that is not familiar with sound levels described in dB. Even if a future legislation should choose to give the limits in L_{den}, it will probably be easier to predict the actual levels as wind turbine noise can be expected to change only very little between day and night unless it is due to the wind changing.

Considering how often different facilities are being planned, and thus how often the product of this project might be put in use, wind turbines are probably planned more often than highways, railways or airports. The reason for this is, among other things because putting up a wind turbine requires far less work than for instance making a new highway, and also beacuse a limited number of airports, railways and highways are needed. Smaller roads and new houses are also built rather often.

When looking at the outlined possibilities, we arrive at the following possibilities to focus the area of the project:

- Wind turbine noise
- Highway noise
- Noise from new (smaller) roads
- Railroad noise
- House planning in noisy environments
- Airport noise

When considering the above mentioned difficulties in predicting noise levels from the L_{den} levels given in the legislations and the frequency of which the above facilities are being constructed, it

has been decided to limit the focus of this project to wind turbine noise, while still allowing the possibility for future expansions to other areas.

2 Presentation of Noise levels

In this chapter the results of the examination of possibly interesting fields will form the basis of the overall system requirements of the final system noise presentation system.

A short overview of possible hardware platforms for a final product implementation will also be presented.

2.1 Basic System Requirements

In the initial analysis of the possible fields where this project could have relevance, it is clear that in all cases single rating numbers are used to set the limits of which sound levels are permissible. In the cases of traffic, railroad and airport noise, the limits are given as a time weighted average og the averaged spectrums, L_{den}, and since the noise in these cases are by no means constant, it can be rather hard to use these levels to make an "example noise sound", at least without adding a large number of extra variables. In the case of the wind turbine, the noise will not change significantly depending on the time of day, and thus it will be easier to make such an "example noise sound".

In the following, different requirements are found either as a result of the target group of the final product or of the purpose.

2.1.1 Target Group Requirements

In the initial analysis, it was found that the target group of this project will be the individual communes, since they are the ones who can make the final decisions on a project. With this in mind, some basic requirements can be defined:

- The final system should be cost effective
  - There are 98 communes with a total of more than 2500 council members in Denmark
  - Most likely at least a handfull of the systems should be present in each commune for the noise evaluation to be effective

  - A cheaper system might be easier to convince people to use

- It should not be possible to tamper with the system
  - The system needs to be realiable and trustworthy
  - Manipulation with results (eg. different output level than stated) should not be possible
- The software should be upgradeable
  - Later improvements, bugfixes and expansions should be possible
- The system should be easy to use
  - The muncipal councils should be able to evaluate the noise without expert knowledge
- Option for linking multiple systems together to play the same sound simultaneously
  - Enables the possibility for one person presenting (using the web interface and possibly a projector) while a lot of people listens to the sample
  - Alternately a method for splitting the output while still keeping the same sound level could be examined

2.1.2 Purpose Requirements

The purpose the project is to aid the decision makers in taking well informed decisions regarding new facilities which may have noise problems. The means to achieve this is to present the sort of limits that are given by law in combination with some explanatory text and/or graphics and a way of presenting the noise acoustically.

Considering the purpose along with the initial examination of noise fields, the following variables should be displayed and some of them be possible for the user to manipulate. Some of the variables are only relevant for some of the noise sources and some are not necessary, but would be nice to have. The latter are denoted as "optional".

- Law limits of given noise (show and illustrate)
- $L_{A,eq(,T)}$ (select from predetermined values)
- L_{den} (show – select in "expert mode")
  - Day, evening and night levels might be relevant to show and edit – especially in other cases than wind turbine noise
  - For wind turbine noise, day, evening and night levels are assumed to be identical
- Distance in m from noise source (select)
- Wind noise: Weather conditions of recording (show)
- Wind noise optional: wind speed – choose 6 or 8 m/s
- Wind noise optional: Angle from wind direction (requires directional recordings)
- Road, railroad an airport noise: Number and distribution of vehicles, trains or airplanes

  - Along with the day, evening and night levels, it might be possible to "synthesize" examples of the noise in the three cases and compare to the noise limits given by law

A nice feature of the system would also be the option to save, load and transfer presets, that is, a specific set of parameters that can be loaded at a later time to present the exact same noise sample again. This would also mean that one person could set up an example that could then be emailed to another person who has a similar system and make sure the same sound is played.

2.2 Hardware Platforms

Looking at the above, the following requirements can be defined for the hardware needed for the system to work:

- A headphone output with full programmable volume control
- An interface for presenting and controlling the device
- Storage for the sound samples
- Optionally a method of communication between multiple devices
- A way to ensure that tampering is not possible either through hardware or software
- As cheap as possible

In the following, an overview of possible hardware platforms are examined according to the above requirements.

LiaB

From the Danish company LiaB (Linux in a Box), the product "nanoLIAB" is a small computer on a single print board, with a complete Linux distribution already installed. The Project relevant features of the system are:

- Small mini pc board (68×62 mm) w/ Linux
- 1w power consumption – allows for battery power
- D/A converter and headphone amplifier
- Plays mp3 files out of the box – The company suggests using it for a small media center or MP3 player
- No direct display support
- A Local Area Network (LAN) interface allows an external to be used for display, control, communication – possibly via a web interface
- 16 MB flash ROM probably doesn't provide sufficient storage capacity
- Universal Serial Bus (USB) interface allows use of USB storage easily

More information, including schematics can be found at the LiaB website[1]. The price of the device is not easily found, but according to a private blog[2] the price is 750 + VAT, ending at a total price below 1000 DKK.

Gumstix

From the company Gumstix, which specializes in small "gumstick size" computers mainly for robotics, a few options are available.

The Overo Water COM[3] along with a Tobi Expansion [4] provides plenty of options for both sound and graphics with openGL via HDMI, DVI-D and communication using ethernet and USB. This package is rather costly (approximately 1300 DKK + cables etc.) and is probably a bit overkill for the current application

The Overo Summit Pack[5] with an Overo Earth COM and a Summit board includes graphics and sound (no openGL) and utilizes micro SD cards for storage. This package (approximately 1400 DKK) comes with a power supply, cables and a micro SD card, but Ethernet is not a possibility with this package.

Both of the two gumstix packages will be even more expensive than the above mentioned prices, as postage from USA and customs duty will apply.

FitPC

The FitPC is a small all in one PC, available in multiple configurations

All fit-PC2i models have the following standard features:

- DVI up to 1920×1200
- Dual Gigabit Ethernet
- S/PDIF 5.1 channels, stereo line-out, line-in and mic
- 4 USB 2.0 ports (2 mini-USB ports on the front with adapters to USB-A included)
- mini-SD slot
- auto-on
- Secured power jack

More information can be found on the company website[6] and the system can be bought from the Danish webshop LinuxShoppen[7] for 2620 DKK the cheapest models.

A system built on the FitPC will definitely work, but will probably be overkilled, for the current application. The advantage of using it, would be that the system can be implemented on any PC system, following the same guidelines.

[1] http://liab.dk/produkter/nanoliab
[2] http://soeren-hansen.dk/2006/11/25/nu-med-ny-hjemmeside/
[3] http://www.gumstix.com/store/catalog/product_info.php?products_id=228
[4] http://www.gumstix.com/store/catalog/product_info.php?products_id=230
[5] http://www.gumstix.com/store/catalog/product_info.php?products_id=220
[6] http://www.fit-pc.com
[7] http://www.linuxshoppen.dk/fit-pc.php

Figure 2.1: nanoLIAB board with a set of headphones. Image is property of LIAB Aps
`http://liab.dk/produkter/nanoliab`

Other Platforms

Two final platforms that could be used for the final product implementation is using a normal laptop, which would require a solid procedure for calibrating and controlling the sound card, or using an "off the shelf" MP3 player, which would also require calibration and (manual) control of the volume settings.

2.2.1 Hardware Decisions

Looking at the features of the above hardware solutions, they all fulfill the basic requirements of sound output and options for making an interface for display and control.

If choosing the "off the shelf" MP3 player, the laptop or the FitPC, the sound might not too easy to control fully without the user interfering and thereby tampering with the device. Looking at the price, neither of the pc's are in the cheapest range.

When comparing the LiaB and Gumstix solutions, the main issue is whether the display and control interface should be directly from and to the device, or through a web interface. Considering the tamper-issue, using a web interface for all display and control seems like a good solution. The price of the device itself will also be cheaper when choosing the LiaB, although it will require a separate computer for the control and display, however it can be assumed that this will not be a problem seeing as most people today have at least one computer at their disposal.

With the above considerations, the nanoLIAB has been chosen as the hardware platform for this project. An image of the nanoLIAB can be seen on Figure 2.1.

3

Decisions and Problem Definition

This chapter is intended to sum up the decisions of the previous chapters as well as to formulate the final problem statement.

After examining the fields of potential interest for the project, it was decided to limit the focus of this project to wind turbine noise, while still allowing the possibility for future expansions to other areas such as traffic, railway or airport noise.

With this in mind, a set of requirements for the system was determined regarding which variables to present and how to present the noise.

To sum up, the requirements of the presentation (which variables to show and how) are as follows:

Presentation requirements

- Law limits of given noise (show and illustrate)
- $L_{A,eq(,T)}$ (select from predetermined values)
- L_{den} (show – select in "expert mode")
  - Day, evening and night levels might be relevant to show and edit – especially in other cases than wind turbine noise
  - For wind turbine noise, we assume day, evening and night levels to be identical
- Distance from the noise source given in meters (select)
- Wind noise: Weather conditions of recording (show)
- Wind noise, optional: wind speed – choose 6 or 8 m/s
- Wind noise, optional: Angle from wind direction (requires directional recordings)
- Road, railroad an airport noise: Number and distribution of vehicles, trains or airplanes

- Along with the day, evening and night levels, it might be possible to "synthesize" examples of the noise in the three cases and compare to the noise limits given by law

By looking at the system requirements, requirements for the hardware platform for the presentation system was set up.

From these requirements a number of possible hardware platforms were examined with evaluations of how well they were fitted for the requirements. The examination led to the choice of the nanoLIAB board as the best suited platform.

3.1 Problem Definition

This section is intended to fully define the problem, now that a number of basic information is known and the and to clarify the focus of the project.

The main goal of the project is to create a device prototype, that will aid public decision makers in taking well informed decisions regarding noise on a given project – in this case focusing on wind turbine noise.

Given this goal, there are three main focus areas of the project:

1. Designing and constructing the device
2. Measuring and recording wind turbine noise
3. Processing the data to enable presentation on the device

The practical problem, now that the basics of some relevant noise fields are covered, will be to construct a relatively cheap device that can present the noise realistically. The basic requirements for the device are already given in the initial analysis This device and the software on it should provide a noise presentation platform with an API that will enable any type of noise to be presented, as long as it is within the limits of the system. The design and construction of the device is documented in *Part IV*.

Apart from this physical device, it has been decided to focus the project on wind turbine noise, and thus measurements and recordings of wind turbine noise as defined in the relevant laws and standards as well as processing of these data will be a main focus of the project. The processed wind turbine noise data should then fit to the API and enable the device to present wind turbine noise to people with little or no knowledge of how this actually sounds. In *Part II* the measurements and recordings as well as the results are described and analyzed and the signal processing to enable noise presentation on the device is designed in *Part III*.

Part II

Field Measurements and Analysis

This part focuses on the measurements necessary to provide data for the noise presentation system.
First, a general overview on wind turbines is given. The general procedure, necessary to adhere to laws and standards, is defined including necessary equipment and considerations.
Following the procedure, the results of a set of field measurements are presented and analyzed. The journal of the measurements can be found in *Appendix A*

4

Measurement Procedure

The following chapter concerns the measurement of noise emitted by wind turbines. Explanations of wind turbines and different sources of wind turbine noise are given. Also, a description of the equipment needed and the procedure for the measurement of the noise according to the current laws and standards in the matter as well as the requirements for a further playback of the recorded noise are presented here.

4.1 Case Study: Wind Turbines

Wind turbines are machines that convert the kinetic energy of wind into electricity. They can be classified differently based on characteristics such as [Age09, Chapter 2],

- Maximum electrical output
- Height from the ground to the top of the blade
- Blade's diameter
- Area covered by the rotor blades

4.1.1 Structure and Types of Wind Turbines

The structure of a wind turbine consists of a rotor that drives a generator, producing electricity. Both the rotor and the generator are installed in the nacelle which is placed on top of the tower leant on the ground. The height of the tower ranges normally from 40 to 160 meters whereas the rotor diameters varies from 35 to 130 meters. The blades of the turbines are always twisted in order to achieve an optimal angle of attack throughout the length of the blade [Ass03a]. The schematics of a typical wind turbine are shown on *Figure 4.1*. Wind turbines can also be classified according to their axis disposition and wind orientation [Ass03b].

- Horizontal axis wind turbines
  Their main rotor and hub are placed at the top of the tower and point the wind to prevent turbulency. Most of the current wind turbines mounted nowadays are horizontal axis since they solve the disadvantages that involve the use of vertical axis wind turbines

- Vertical axis wind turbines
  The main rotor is disposed vertically. The main advantage is that the efficiency of the wind turbine does not depend on the wind direction making them suitable for places where the wind is remarkably variable. The main drawback of this turbines is the difficulty to implement them on towers. This involves that the axis is placed close to the base of the wind mill (e.g. the ground) and the height of the mill short. The wind speed is slower at low heights being as a result a less amount of energy produced

- Upwind turbines
  Upwind turbines face the wind. Most of the turbines are built upwind since the wind shade behind the tower is then avoided. On the contrary, these turbines need a rotational mechanism in the tower to follow the wind direction and the rotor needs to be built at a certain distance from the tower

- Downwind turbines
  Downwind turbines are placed on the lee side of the tower. Their main advantages are that a rotational axis to follow the wind is not needed and that the weight distribution achieved for this configuration makes them lighter than the upwind turbines. On the other hand, inconveniences presented by this placement of the turbine is that the design is much more complex to solve the fatigue loads appearing in the turbine

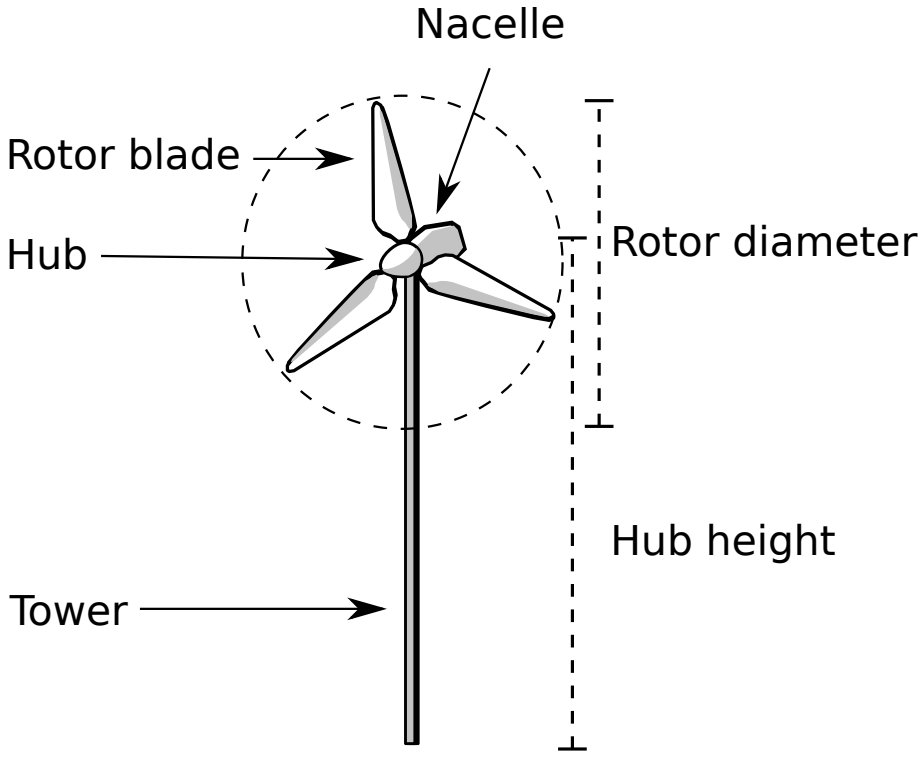

Figure 4.1: Sketch of a horizontal axis wind turbine

4.1.2 Sources and types of Wind Turbine Noise

As a result of the operation of a wind turbine, four types of noise can be identified [ALR04],

- Tonal noise, defined as noise at discrete frequencies
- Broadband noise, characterized by a continuous distribution of sound pressure for frequencies greater than 100 Hz
- Low frequency noise, localized in the range of 20 to 100 Hz
- Impulsive noise, varying the amplitude in time

These four different types of noise are present on wind turbines depending on the sources that produce them. The sources of noise from a wind turbine are mechanical and aerodynamic.

- Mechanical Noise. Generally, it is a tonal noise which may have a broadband component. It is created by the motion of the mechanical components placed mainly in the nacelle such as the gearbox or the generator, and radiated by the tower and the rotor
- Aerodynamic Noise. This noise has a broadband characteristic and is produced by the air flow around the blades. The higher the rotor speed is, the more noise will result

4.2 Wind Turbine Noise Measurement Guide

The Announcement no. 1518 from the Danish Ministry of the Environment [Mil06a] refers to the standard IEC 61400-11, *Acoustic Noise Measurement Techniques on Wind Turbine Generator Systems*. Measurement requirements given by the Announcement are described in detail in *Appendix A* and all formulas and corrections are listed. The aim of this section is to serve as a quick guide to setup all necessary equipment for a correct measure of wind turbine noise.

4.2.1 Measurement Equipment

The following explains how to set up the equipment and proceed for the measurement of wind turbine noise. In order to measure and record wind turbine noise, the equipment needed is listed on *Table 4.1*.

4.2.2 Weather Station Set-up and Operation

When measuring wind turbine noise, the simultaneous measurement of the wind speed and its direction, the pressure, the temperature and the humidity on the field is essential to fullfil the requirement of the standard IEC 61400 and to ensure a correct functioning of the equipment. For that purpose, a weather station is used, providing all information required via wireless connection. The built-in anemometer of the weather station comes along with the software Heavy Weather 2.0 which can to collect all related weather data. The program has an option for saving the data of the measurements in a text file that can be read in Matlab with a function called *weatherread.m* that can be found on the CD attached. This function has been implemented for such purpose in order to enable further processing.

Item	AAU No.
B&K 4165 1/2", free field microphone	07954
B&K 4165 1/2", free field microphone	07955
B&K 2238 Mediator, sound level meter	33948
B&K 4230, calibrator for B&K 4165	08373
B&K 4231, calibrator for B&K 2238	78301
Grass 26AK, preamplifier	52664
Grass 26AK, preamplifier	52665
Roland-Edirol R-09, Digital recorder	64676
01 dB Harmonie, data acquisition system	04124
Laptop HP Omnibook 6000	47220
Weather station with anemometer	2157-45
dB Trig32, data acquisition software 01 dB	
Base plate	
LEMO, connection cables	
1/2 wind screen	
10 m. high mast	

Table 4.1: Equipment needed for the measure of wind turbine noise

1. Prior to any measurement, wind data must be known so that a plate with a microphone can be placed correctly. The weather station has an anemometer that has to be fixed to the mast. It is recommended to check that the anemometer works properly before mounting it on the mast. For that, the next steps shall be followed.

   (a) The weather station measures its surrounding atmosphere and receives the outer weather conditions by means of a exterior unit which incorporates the following 3 sensors,

   - Thermo-Hygro sensor: measures all relevant information regarding atmospheric conditions (temperature, pressure and humidity)
   - Wind sensor: collects all relevant information regarding wind conditions (speed and direction)
   - Rain sensor: measures all relevant information regarding rainfall data (1 hour, 24 hours estimation)

   The Thermo-Hygro external sensor is the main communication unit since both the wind and the rain sensors are connected to it. Besides, the thermo-hygro sensor acts as a power supply for the other 2 sensors

   (b) All received data is updated continuously. Thus, the displayed weather conditions in the weather station are as recent as possible. The sensor unit has 3 plugs (see *Figure 4.3*) that receive and transmit the acquired data to the base station. The plug in the left receives the rainfall data and the one in the right the wind data. The plug in the middle connects the unit to the station via a provided serial cable and can be identified by the word "display". The provided cable is only 10 meters long and taking into account that the mast is at least 10 meters high, the cable is not useful for the desired purposes unless the working position is directly next to the mast. Then, the wireless options seems to be more appropriate. Nothing special has to be done to setup a wireless configuration. Simply, by not connecting the cable to the "display" plug on the sensor unit, the wireless mode is automatically activated. It has a range of 100 meters on a free path

   (c) The next step is to set up the weather station. For that, connect the serial port cable from the weather station base plug "PC COM" to the computer "COM" port or with an "USB" adaptor (see *Figure 4.4*)

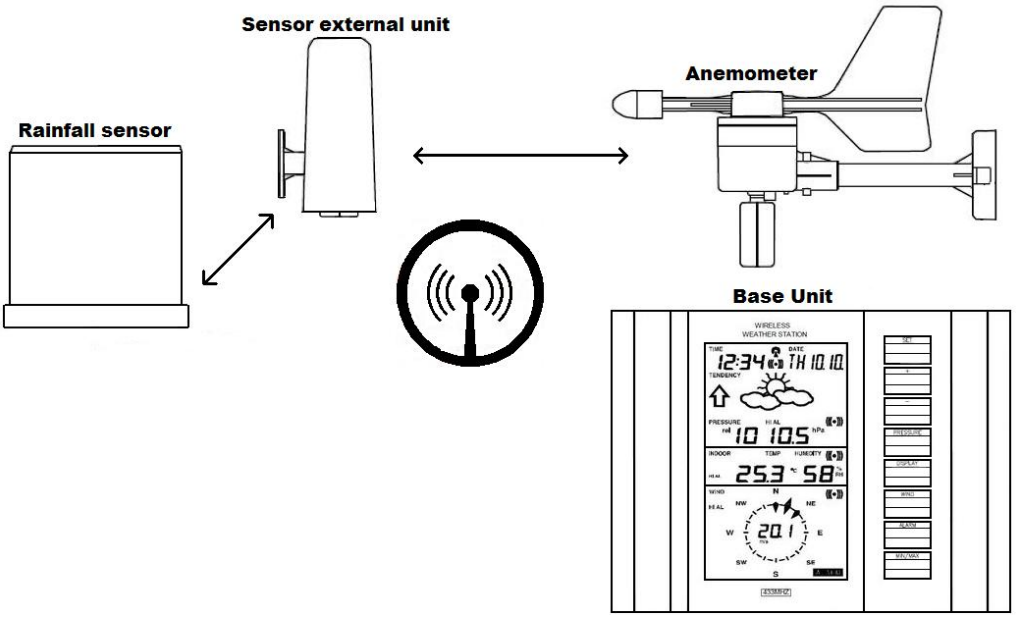

Figure 4.2: Connection anemometer-sensor-weather station [Weaa, page 1]

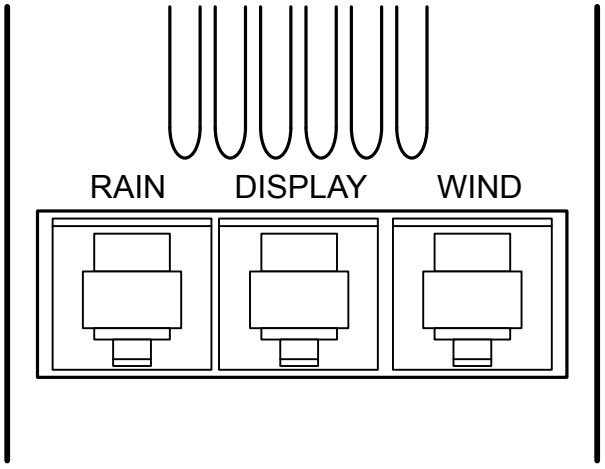

Figure 4.3: External unit sensor [Weab, page 1]

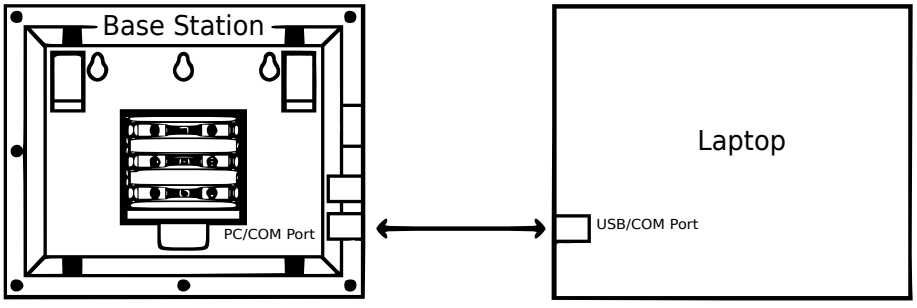

Figure 4.4: Connection weather station base - computer

(d) Run the software Heavy Weather 2.0 installed on the computer and click on "setup". The tab global sets the com port that is being used and can be changed. The tabs units and pressure are intended to set the desired units for the measurement. *Figure 4.5* shows a typical screenshot of the setup window

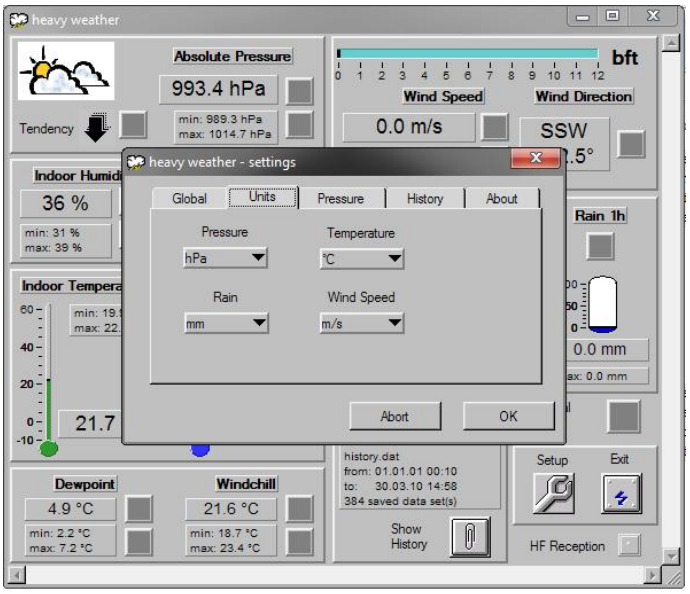

Figure 4.5: Heavy Weather 2.0 setup options

(e) Once the settings have been applied, the weather station is almost ready to measure. Before starting to measure via wireless, the sensors have to be reset in order to synchronize them properly. For that, press the button *PLUS* (*"+"*) for two seconds, until a *"bip"* is heard. This key can be found opening the cap on the lower part of the station

(f) Finally, check that wind speed, wind direction, ambient pressure, temperature and humidity are being collected. Wind speed and direction are key for the positioning of the plate where the microphone is placed. Ambient pressure, temperature and humidity are useful to ensure a correct performance of the rest of the equipment since extreme condition such as very high temperatures may damage it. It is important to remark that, in the wireless mode the maximum range of operation is 100 meters. Furthermore, wind speed and direction are not collected instantaneously. The sensor averages all collected data during 2 minutes that can lead to the idea of a wrong configuration. *Figure 4.6* shows a typical screenshot of the program running properly

2. The adequate position where the mast with the anemometer can be placed must be known (*see page 118 of Appendix A*) Once this is known, install the anemometer onto the mast. The height of the mast must be at least 10 m. Place it on the previously calculated position (which must be on the upwind direction of the tower) and check that data collection progresses as expected prior to the measurements

4.2.3 Wind Turbine Noise Measurement

For the measurement of the noise produced by the wind turbine, the process is divided in 3 different sections. The first one concerns to the set-up chain needed for the measurement, in which a discussion about the type of microphone used and the procedure to set it up to the hardware

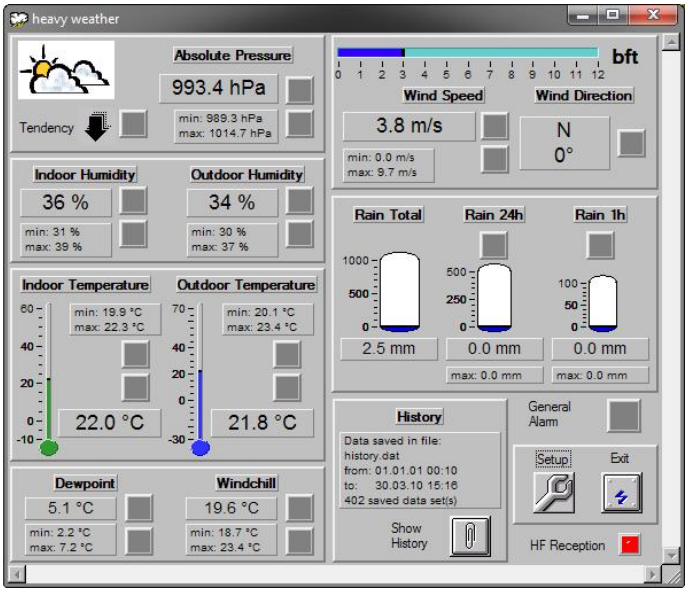

Figure 4.6: Operation of the software Heavy Weather

platform Harmonie is included. The second part concerns the configuration of the analysis to be performed with the software provided along with the Harmonie system, the software dBTrig 32. Lastly, the third part describes how to measure the background noise where a guide of how to use the sound level meter B&K 2238 and the digital recorder Roland Edirol is provided.

Microphone Set-up

To determine the equivalent continuous A-weighted sound pressure level (L_{Aeq}), the used microphones must be of a diameter no greater than 13 mm. according to the standard IEC 60804 [Com00] and mentioned in the standard IEC 61400. However, no specification of the type of microphone is given.

In [BK04], Brüel & Kjær performs a measurement of wind turbine noise similar to what is desired in this thesis, recommending the use of one of their free field microphones. The type of microphone to use may lead to an interesting discussion wether to use a free field or a pressure response since the microphone is not mounted on a stand (is not "free") but on a plate placed on the ground. The plate is flat and acoustically hard, according to the standard.

A free field microphone is designed to capture sound waves which propagate free in a continuous medium like the air. This would be the case if the microphone is placed on a stand at a certain height but not when placing it on the ground next to a robust surface such as the plate. On the other hand, a pressure field is used when the sound pressure has the same magnitude and phase at any position. This is not the case either. For frequencies below 5 kHz, 1/2" microphones behave very similar and the output given might vary $\pm 2dB$ [DFTN99]. Therefore, differences reside for higher frequencies. When measuring noise from wind turbines, the microphone is placed on a plane normal to the plate and pointing towards the wind turbine (0°incidence). The frequencies of interest range in octave bands from 63 Hz to 8 kHz. If a pressure response microphone is used, the level registered by the microphone will be higher as the frequency grows. This is because the incoming wavelengths are smaller producing an amplification on the diaphragm due to refraction and reflection of the waves. However, if the microphone used is a free field, the high frequency pressure amplification at 0°is canceled, providing better results. For this reason the used microphones are

free field type.

For a correct setup and measurement of the wind turbine noise the next steps should be followed:

1. Connect the microphone to the preamplifier and connect it to one of the inputs of the Harmonie system with a LEMO cable as shown on *Figure 4.7* The Harmonie system must be connected to a laptop with the corresponding software installed. A deeper description of the Harmonie system is provided in *Section 4.2.4*. Run the program dB Trig 32 and calibrate the microphone with the calibrator and the software prior to any measurement. To calibrate the microphone follow the next steps and illustrations

   (a) Connect the Harmonie system to the computer as in the diagram of *Figure 4.7*

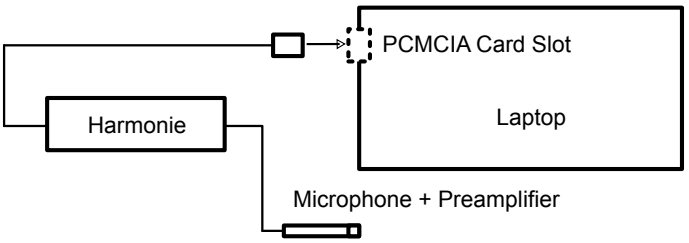

Figure 4.7: Setup of the Harmonie system

   (b) Prior to any measurement, a calibration of the equipment is a must. For that, after running the program dB Trig 32, the microphones and calibrators to be used have to be added. For that, click on *Setup* → *Hardware configuration* and a window as the one in *Figure 4.8* will pop up. Click on "Transducer" and "Calibrator" to add a new device to the system if necessary

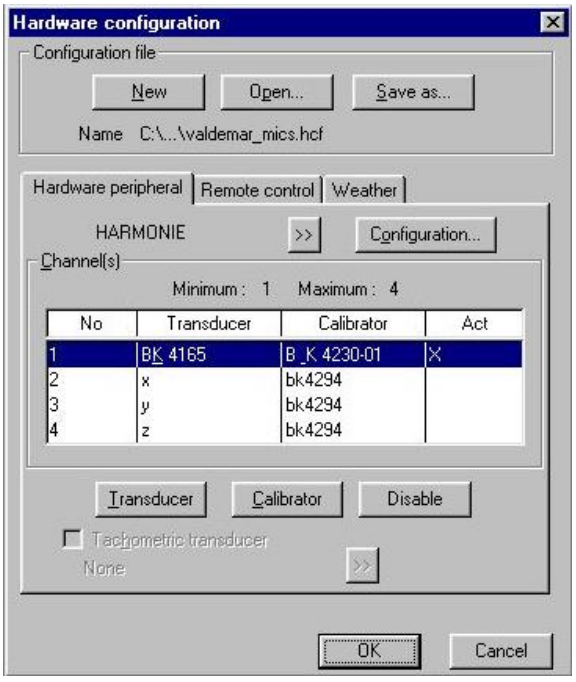

Figure 4.8: Adding equipment to the Harmonie system

   (c) After adding the new equipment, a calibration can be performed. Clicking on *Setup* → *Calibration* opens the calibration window. Select the equipment to be calibrated and

click “Execute” after inserting the microphone in the calibrator shown as on *Figure 4.9*

Figure 4.9: Microphone calibration procedure

(d) A new window will open (see *Figure 4.10*) in which the button “Adjust” must be clicked to set the microphone to the 94 dB reference. Finally, click on “Valid”

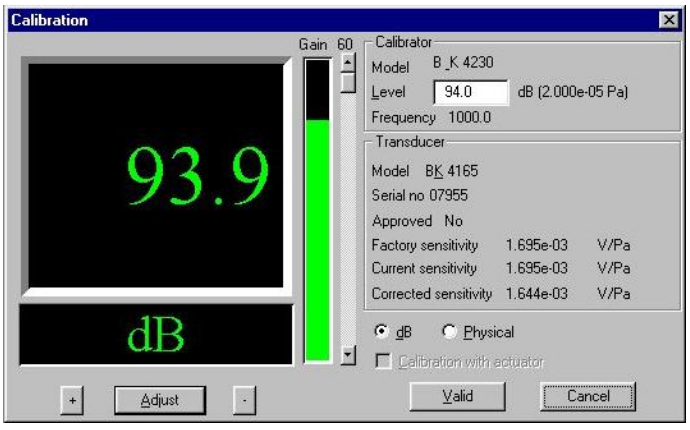

Figure 4.10: Calibration of the equipment used

2. Once the microphone is calibrated, it must be placed on the plate with a half wind shield to avoid noise from the wind. Place the plate on the ground at a distance equal to the height of the tower plus half of the diameter of the blades (*see page 118 on Appendix A* for a complete description). The microphone must be pointing towards the tower of the wind turbine in the direction of the wind

4.2.4 dB Trig Measurement Setup

Wind turbine noise data is collected by means of the Harmonie measurement system. Harmonie, with the 01dB software dB Trig32, combine the functions of a sound level meter, a digital recorder and a real time frequency analyzer. For further data processing, measurements can be saved and imported to Matlab by using the function *harmonie2mat* attached in the CD.

1. In the dB Trig 32 program, click on *Setup* $\rightarrow$ *New* to start a new L_{eq} measurement, the menu on the left is essential to setup the measurement (see *Figure 4.11*) To perform a L_{eq}

Figure 4.11: dB Trig 32 measurement setup

measurement, follow the next steps,

(a) In the parameters window (see *Figure 4.12*) set up measurement pass band, time base and frequency weighting (Linear and A-weighting in this case) desired by clicking on the button "Measurement"

(b) By clicking on "Storage", information about the reporting of the data can be entered and the duration of the measurement be programmed

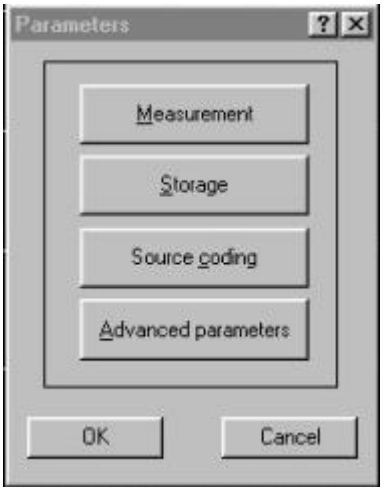

Figure 4.12: dB Trig 32 options

(c) Starting the measurement can either be done by clicking the "rec"button on the menu or by pressing "F3"

2. The recording can also be done with the same microphone as the noise measurement with the Harmonie system. To start the recording press the "cassette" button on the bottom part of the left menu of the program (see *Figure 4.11*)

4.2.5 Background Noise Measurement

According to the standard IEC 61400, the background noise must be measured and recorded in the exact same position as when measuring the wind turbine noise. This involves that the wind turbine had to be stopped and then, the background noise measured. Unfortunately, it was not possible to stop the turbine and the background noise could not be measured reliably. An alternative method is proposed by means of the estimation of the percentile level, L_{90} (see *Section 5.2.2*). However, since this section is intended to serve as an user guide, a description of the steps to measure as specified by the standard IEC 61400 is given. Procedures for measuring and recording are explained separately.

Set-up of the Sound Level Meter

To measure the background noise it is proposed to use the sound level meter B&K 2238 Mediator. The following steps explain how to set up the sound level meter to do a proper measure.

1. To measure the noise with the sound level meter, first thing to do is configure the analysis [BK02, Chapter 3]. For that, follow these steps:
   (a) Press the "Power" button placed on top of the sound level meter panel
   (b) Once the sound level meter is turned on, press the button "System" to enter the menu
   (c) In the menu, select "application" with the button "Select" and check that "Frequency Analysis" is selected. This analysis consists on a number of consecutive measurements performed for each of the selected bands. The result is a spectrum measured over a desired time period
   (d) Press "Meas. Controls" and select the frequency weighting in the menu
   (e) The bandwidth can be set either to "1/1 Octave" and frequency limits "31.5Hz-8kHz" or to "1/3 octaves" and frequency limits "50Hz-10kHz"
   (f) Check that the broadband weighting is set to "A"
   (g) Press "Ok" and go to the previous menu
   (h) Back in the menu, select the option "Meas. Control". In this section it can be set the number of scans (10 according to the standard IEC 61400), the dwell time ("optimized" with a tolerance of 0.5 dB) and the "Auto Save" option which is recommended to be active
   (i) It is recommended to save the setup after the configuration Back in the menu, navigate downwards and select "Save Setup"
   (j) Press "Close" to go to the main screen (see *Figure 4.13*) and press the button "Range" to select the dynamic range. Set it into the range between 20-100 dB and press "Ok"
2. After having the measurement configured, the next step would be to calibrate the sound level meter. This is an essential action that must be done prior to any measurement [BK02, Chapter 4]. For that, it is necessary to use the B&K 4230 calibrator and follow the next steps displayed on the sound level meter
3. Once the setup is configured and the calibration performed, the measurements can be carried out. For that, cover the microphone with the wind shield, place the sound level meter on the plate and press the "Play/Continue" button. To check the progress of the measurement, press the button "SLM" to change from one screen to another. Thus, the remaining time and the number of scans left can be observed

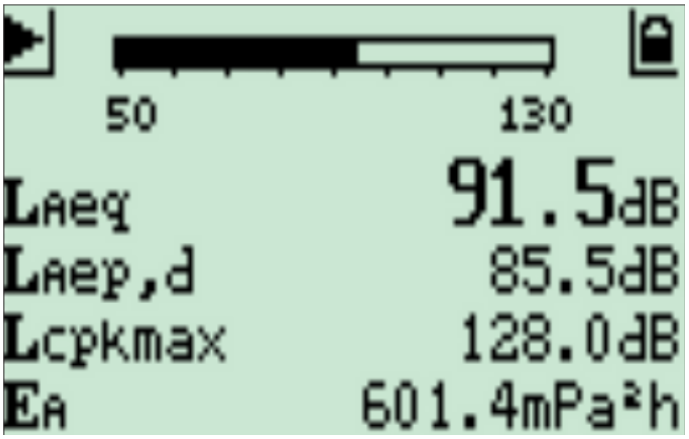

Figure 4.13: Display of the B&K 2238 Mediator

4. When the measurement is over, data is saved automatically when the "Auto Save" option is enabled. Otherwise, it can be stored manually by pressing the button "Data Files" [BK02, Chapter 5]. A file number will be displayed indicating the position of the stored data in the saving list

Set-up of the Digital Recorder

The digital recorder proposed to record the background noise is the Roland Edirol R09. The digital recorder can be difficult to calibrate and it is proposed to do so by recording a calibration signal with it, find a correction factor and estimate the recorded levels as it is done for the data acquired with the Harmonie system (see *Section 5.2*). The digital recorder Roland Edirol R09 allows the recording of sounds in both WAVE and MP3 formats, storing the recorded data into a SD-Card. The sample rates allowed are 44.1kHz and 48kHz. To ease the level estimation, it is recommended to record the calibration signal and the background noise recordings with the same sample rate. Otherwise, resampling would be necessary for either the calibration signal or the noise signal. The functioning of the recorder consists on the classical pressing of the buttons "record" and "stop'" The resulting audio file is automatically saved.

5 Results

Measurements were carried out in the proximity of Aalborg. The corresponding measurement journal can be found in *Appendix B*. Measurements were carried out according to the rules established by the standard IEC 61400 as much as possible. The following limitations should be taken into account for the evaluation of this set of measurements.

5.1 Considerations and Limitations

1. The measurements consisted of nineteen periods of one minute concurrent with measurements of the wind speed and not the thirty required by the standard. The location of the wind turbines within a farmland limited the measurement time due to agricultural activity on site and thus such requirement could not be completely fulfilled. Measurements were carried out at a distance of 51.8 meters away in the direction of the turbine with a deviation of no more than $\pm 15^{\circ}$ from the direction of the wind.

2. The wind conditions present at the time of the measurements were not as satisfactory as desired. Wind speeds did not fulfill the most satisfactory conditions resulting into very slow speeds, i.e. wind speeds lower than 6 m/s

3. It was not possible to carry out the measurement of the background noise at the exact same position since the turbine could not be stopped. Besides, difficulties of finding a quiet place away from the turbines arose, and it was decided to determine the background noise by estimating the percentile level L_{90}.

4. The construction of a wind mast intended to provide a reliable wind conditions measure (minimum height of 10 meters) was delayed more than two months, limiting the time available to perform the measurements. Obtaining a proper mean of transport to carry all equipment to the venue was difficult and thus limited the chances to measure to only one time. Besides, unfavorable weather conditions predominated during the semester in which this master thesis was held. The performance of measurements with weather conditions such as

rain or snow might have put in a very high risk the health of the equipment which is not property of the holders of this thesis but of the university

5.2 Analysis of the results

The following section analyzes the wind turbine noise results obtained in the field. The Harmonie system measures and records the data separately, however in *dBTrig* only the measured data is displayed. An option allowing to export recorded data results in the generation to an audio file in WAVE format. This file will be read in Matlab to validate that measured and recorded signal levels are equivalent. It is important to remark that an accurate estimation of these levels is crucial for a correct playback of the signals by the *nanoLIAB*.

5.2.1 Analysis of the results with Harmonie

Data acquired with the hardware module Harmonie was stored by the software dBTrig. For the subsequent analysis, the results are shown by the software dBTrait. Data acquisition was set up so that Harmonie averages all collected data during one hundred milliseconds. Thus, a total of six hundred samples are collected per minute.

Figure 5.1 plots the A-weighted sound pressure level obtained during the first measurement of the wind turbine noise and *Figure 5.2* plots the whole measurement session, minute by minute. The noise amplitude is comprised mainly between 52 and 55 dB(A) and it includes both the noise generated by the turbine and the background noise. In *Appendix B* can be seen all measurement plots for each wind speed integer.

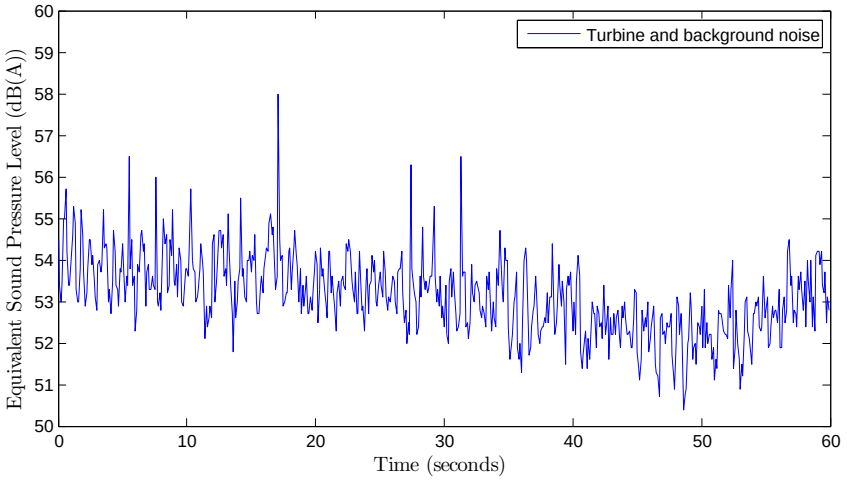

Figure 5.1: Data acquired with Harmonie during 1 minute

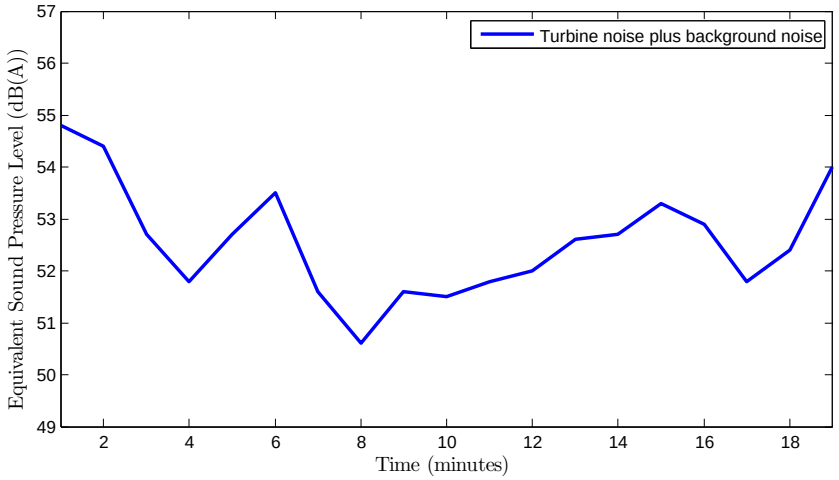

Figure 5.2: Data acquired with Harmonie during the whole measurement session

Data acquired in the frequency domain was done according to specifications given by standard IEC 61400 in third octave bands from 50 Hz to 10 kHz. As it stated in the standard, an average over at least three spectra of one minute for each integer wind speed was performed. *Figure 5.3* shows an example of this arrangement of the data according to the wind speed. These results are expected since wind turbines emit noise in the low-mid frequencies. As it can be seen, most of the spectral information covers the low-mid frequencies from 250 Hz to 2 kHz. The rest of the spectra can be seen in *Appendix B*.

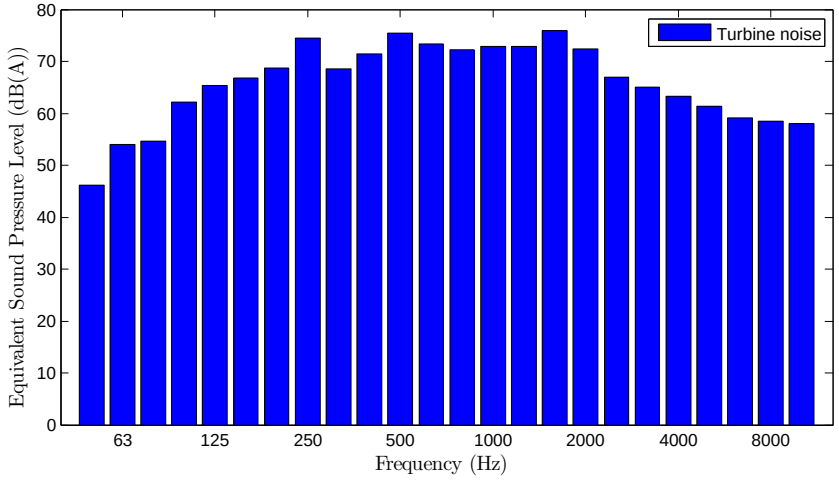

Figure 5.3: 3-minute A-weighted averaged spectra for a wind speed of 3.2 m/s

5.2.2 Analysis of the results with Matlab

In the previous section, it was remarked that Harmonie records and measures separately and that the data displayed by the program dBTrait are the measured sequences but not the recordings. Thus, the purpose of analyzing the recorded results with Matlab is to find the coincidence between the levels measured and the levels recorded. In other words, the data measured with Harmonie,

serve to check that the estimated levels of the WAVE file are correct. For that, a function called *get_levels.m* is developed and included in the CD. The function is structured into three parts. First part calculates a correction factor which is necessary to make a correct estimation of the levels in dB(A) since the WAVE data read in Matlab has no other meaning than a value between 0 and 1. Second part computes the equivalent A-weighted sound pressure level of the recording signal by averaging periods of 100 ms. and 1 minute, as during the acquisition of data with Harmonie. This way, a direct comparison can be done. Third part computes the background noise of the signal, which is needed on a further stage of the data processing to estimate the sound power level of the turbine $L_{wA,ref}$. As commented on *Section 5.1*, it was not possible to determine the background level during the measurement session and then it must be determined by the percentile level L_{90}.

Correction factor calculation

The diagram in *Figure 5.4* shows the calculation process of the correction factor. For that, the signal is read, A-weighted and examined on the frequency domain. The values in dB(A) are calculated for each one-third octave bands from 50 Hz to 10 kHz. Next, the value at 1 kHz is picked and applied in *Equation 5.1* to obtain the correction factor,

$$CorrectionFactor = 10^{\frac{OL-x}{20}} \tag{5.1}$$

where "x" is the value read at 1 kHz after the one-third octave band conversion, "OL" is overall level in dB(A) of the signal measured with Harmonie and thus, taken as reference for the level estimation.

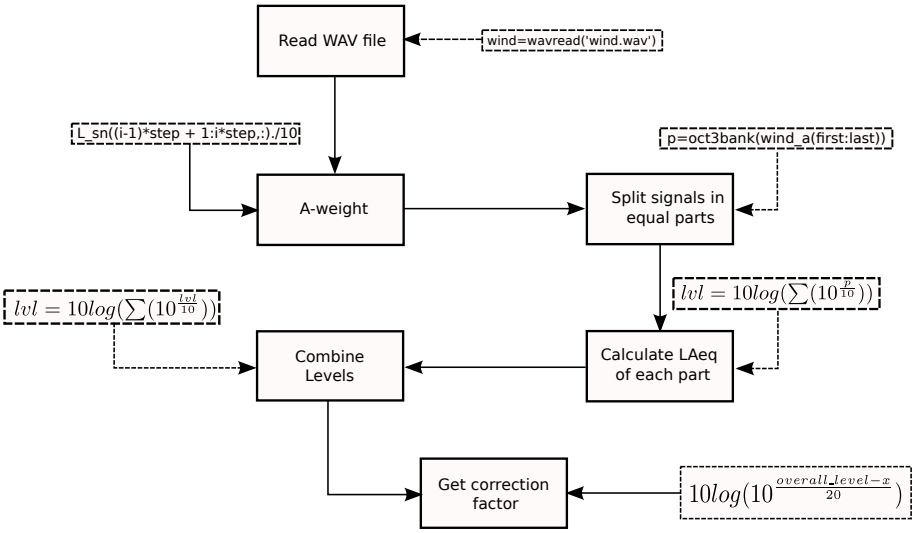

Figure 5.4: Correction factor calculation process

Equivalent A-weighted sound pressure level calculation

The diagram in *Figure 5.5* shows the calculation process of the equivalent A-weighted sound pressure level. The signal is read and each sample multiplied by the correction factor calculated previously. After that, the signal is A-weighted and split up into chunks of 100 ms and 1 minute

using the sampling frequency of Harmonie, 51200 Hz. The signal then can be compared with the data acquired by Harmonie in two different ways. Next, every chunk is filtered in one-third octave bands to calculate the L_{Aeq} of each chunk by means of *Equation 5.2*,

$$L_{Aeq,T_{chunk}} = 10log\Big(\sum 10^{\frac{L_band}{10}}\Big)dB(A) \tag{5.2}$$

where $L_{Aeq,T_{chunk}}$ is the equivalent A-weighted sound pressure level of each signal chunk of length "T" seconds and L_{band} is the level of each one-third octave band on each signal chunk.

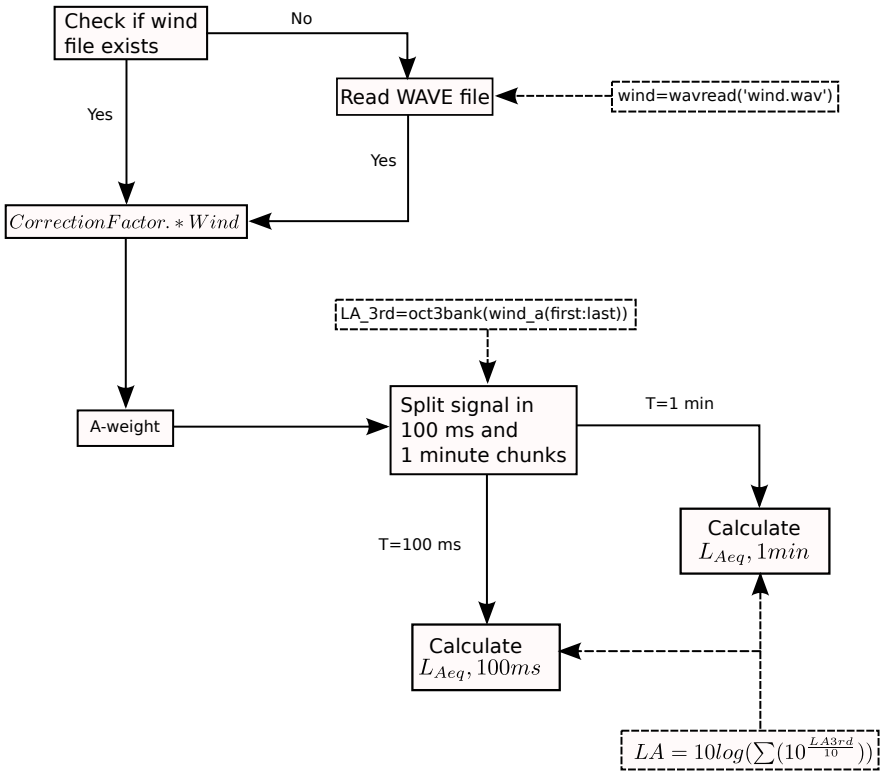

Figure 5.5: A-weighted sound pressure level calculation process

Background noise estimation

The background noise was estimated by means of the percentile level, L_{90} an indicator frequently used to rate industrial noise affecting residential and industrial areas [Gro]. Percentile levels are widely used when measuring environmental noise and range from 1 to 99. This numeric index indicates the % of the signal lasting time in which this level is exceeded. Therefore, L_{90} indicates in this case a level in dB(A) which is exceeded 90% of the time in the signal. *Figure 5.12* represents the background noise estimation by means of the percentile level estimator, L_{90}.

Figures 5.6, 5.7 and *5.8* show the results obtained in Matlab after the execution of the recently explained function *get_levels.m* for the same examples given in *Section 5.2.1.* As can be appreciated in the figures, the results are quite satisfactory when compared with those obtained with Harmonie. *Section 5.2.3* get into more details within this issue.

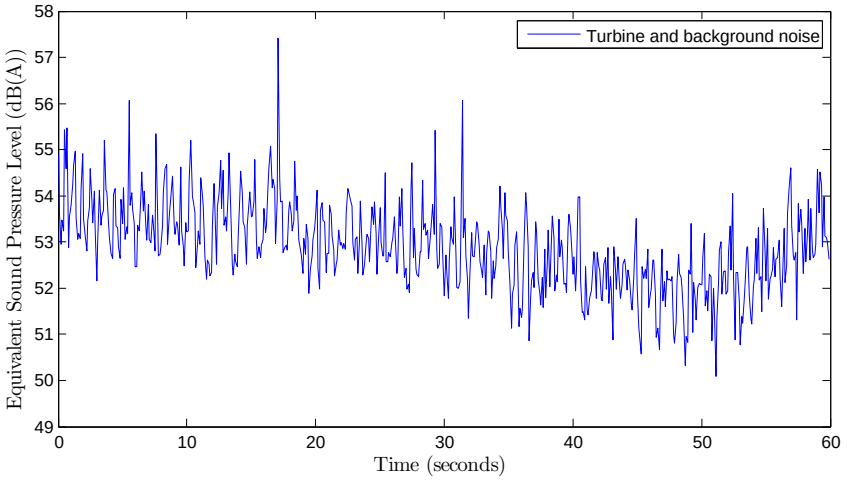

Figure 5.6: WAVE data analyzed with Matlab for a 1 minute signal period

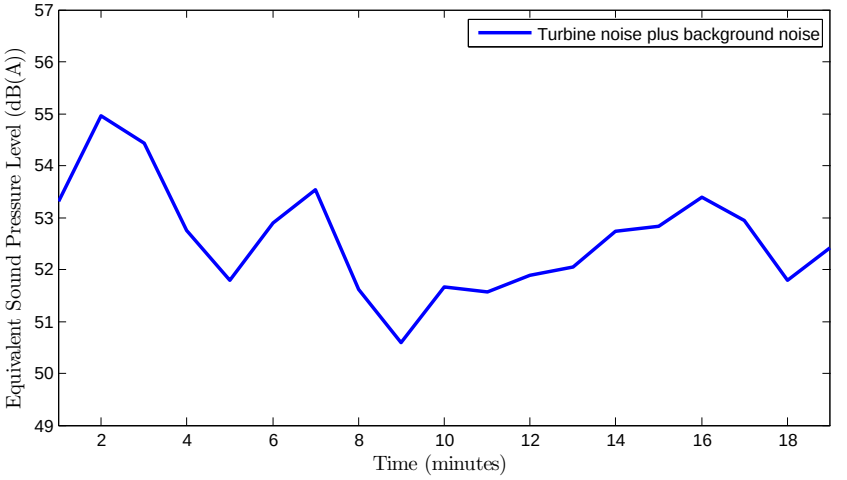

Figure 5.7: WAVE data analyzed with Matlab for the whole measurement session

5.2.3 Validation of results

In the previous two sections, it was explained how the data was acquired with different acquisition platforms and some examples were given. It was previously remarked that the data analyzed obtained in the software dBTrait correspond to the measured data, and not the recorded. This section details how measured and recorded sequences are compared and shows the little differences between them. When analyzing data a synchronization problem between the sequences was observed. The start of the measurement and recording processes in Harmonie is not simultaneous and has to be done manually. As a consequence of this, a larger sequence was observed when comparing the length of measured and recorded data. Thus, initial instants were separated each other by 800 ms. and a clip of this length had to be applied to the measured data sequence to solve the problem. Once this was done, results for each minute average could be directly compared (see *Table 5.1*).

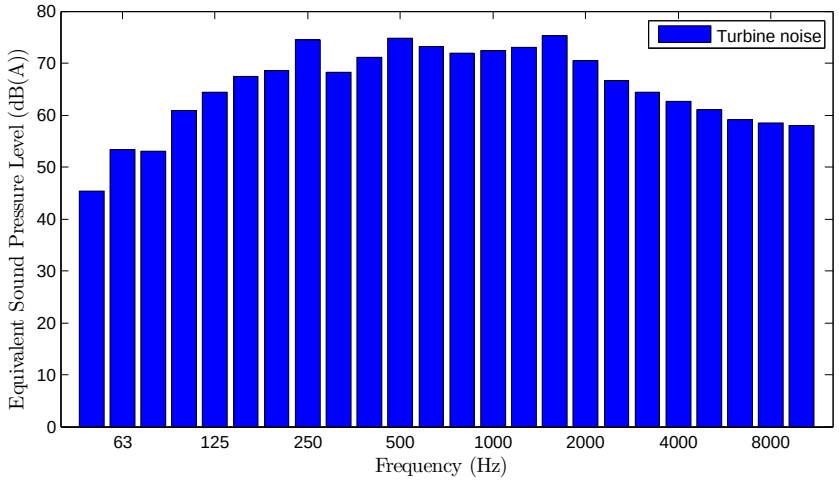

Figure 5.8: 3-minute averaged spectra for a wind speed of 3.2 m/s

Minute no.	$L_{Aeq_{WAV}}$	$L_{Aeq_{Harmonie}}$
1	53.30	53.3
2	54.95	54.8
3	54.41	54.4
4	52.74	52.7
5	51.78	51.8
6	52.88	52.7
7	53.53	53.5
8	51.59	51.6
9	50.58	50.6
10	51.65	51.6
11	51.55	51.5
12	51.87	51.8
13	52.03	52.0
14	52.72	52.6
15	52.82	52.7
16	53.38	53.3
17	52.92	52.9
18	51.78	51.8
19	52.38	52.4

Table 5.1: Comparison between data obtained in measurements with Harmonie and WAVE file data processed in Matlab

Figure 5.9 shows both signals within a twenty seconds interval. Small differences can be observed in the level amplitudes being the Harmonie levels a bit higher, possibly due to the inner functioning of the Harmonie system which might do a previous re-sampling of the signal modifying the gain.

In the frequency domain, a comparison on one-third octave bands was done and shown on *Figure 5.10* for the same cases was given in the previous two sections. This example clearly shows the good performance given by the function *get_levels.mat* as the estimation of the levels on each one-third octave band is turned to be very accurate.

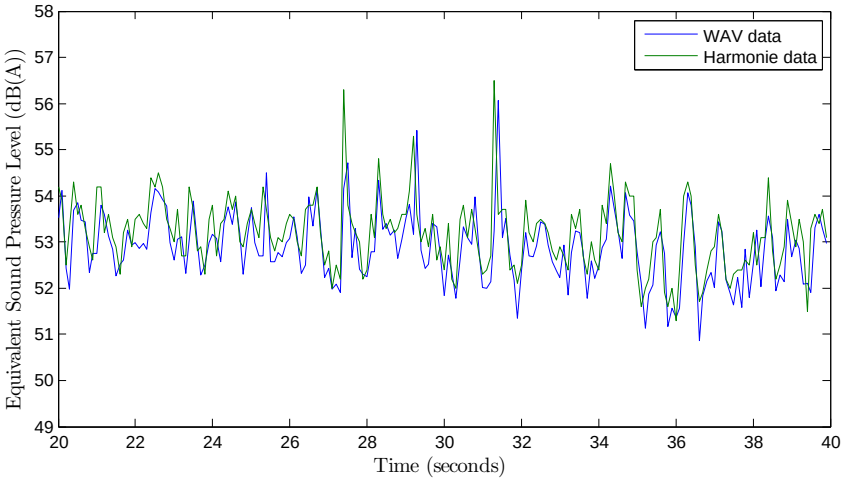

Figure 5.9: WAVE data versus Harmonie data on a 20 seconds interval

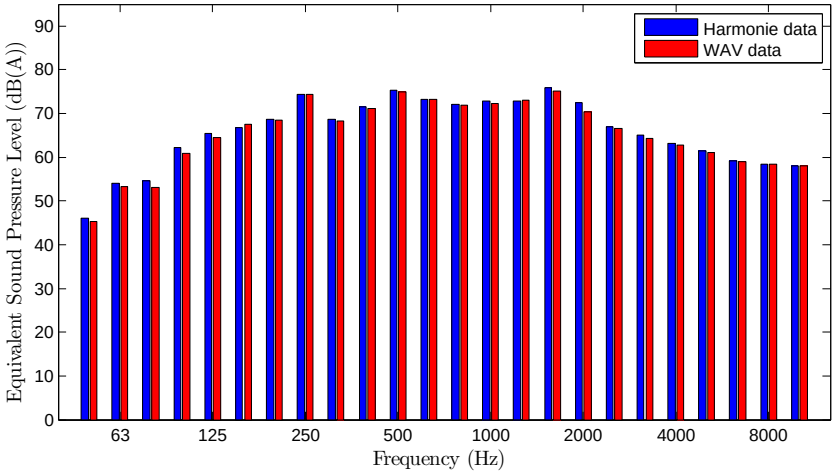

Figure 5.10: 3-minute averaged spectra for a wind speed of 3.2 m/s

5.3 Post-processing of the results

In *Section 5.2.3* the model proposed through Matlab to estimate the levels measured on site by means of an audio file recorded in WAVE format was validated. However, the standard IEC 61400 and the Announcement no. 1518 from the Danish Ministry of the Environment specify further processing of the acquired data. The above mentioned processing consists in applying corrections on both the wind speeds and the levels of the recorded signal. For this purpose *Equations A.2,A.3* and *A.4* specified on *Appendix A* are implemented in a Matlab function called *corrections.m*. The diagram shown in *Figure 5.11* shows the whole data processing consisting of a normalization of the measured wind speeds, a correction for the estimated background noise and the final calculation of the apparent sound power level of the turbine in one-third octave bands. The next step in the processing of the data is to determine the sound pressure level at a given distance for the set of wind speeds measured.

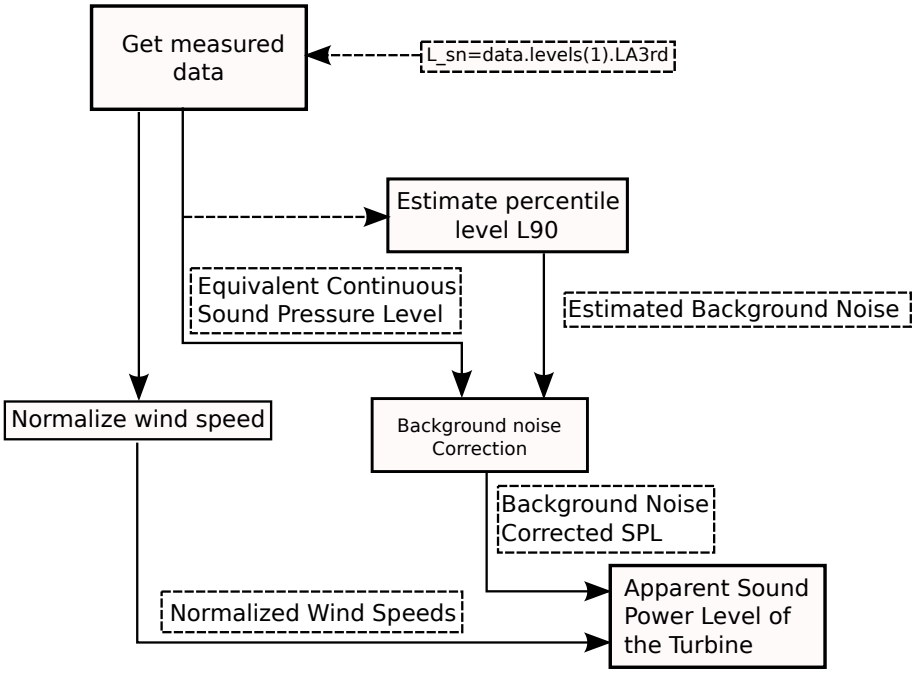

Figure 5.11: Correction guidelines specified by the standard IEC 61400

5.3.1 Wind speed normalization

The ideal method for an adequate normalization of wind speeds to a reference height of 10 m. requires to know the instantaneous power of the turbine and its power output curve. Unfortunately, this data was unknown due to the antiquity of the measured turbine. However, the standard IEC 61400 foresees this fact proposing an alternative method for the normalization. This method considers the type of terrain in which the measurements are carried out and applies different correction regarding the terrain. This was the method followed during the measurements although, as the measurement height was the same as the reference of 10 m., new wind speed values turned to be the same.

5.3.2 Correction for background noise

Figure 5.12 indicates the calculation process followed to apply the background noise correction proposed by the standard IEC 61400. Data obtained after the execution of the function *get_levels.m* is organized in periods of one minute and treated in one-third octave frequency bands. For the background noise computation, as it could not be measured on site, the percentile level L90 is computed for each one-third octave frequency band. For a further accurate calculation of the apparent sound power level of the turbine, a discrimination is applied to the values with a difference with the background noise no greater than 6 dB. *Figure 5.13* shows the equivalent continuous sound pressure level of the wind turbine plus the noise for a one minute period in one-third octave bands and the estimated background noise. *Figure 5.14* shows the result of the correction in which the levels are practically kept constant due to the high differences between recorded the levels and the estimated background noise.

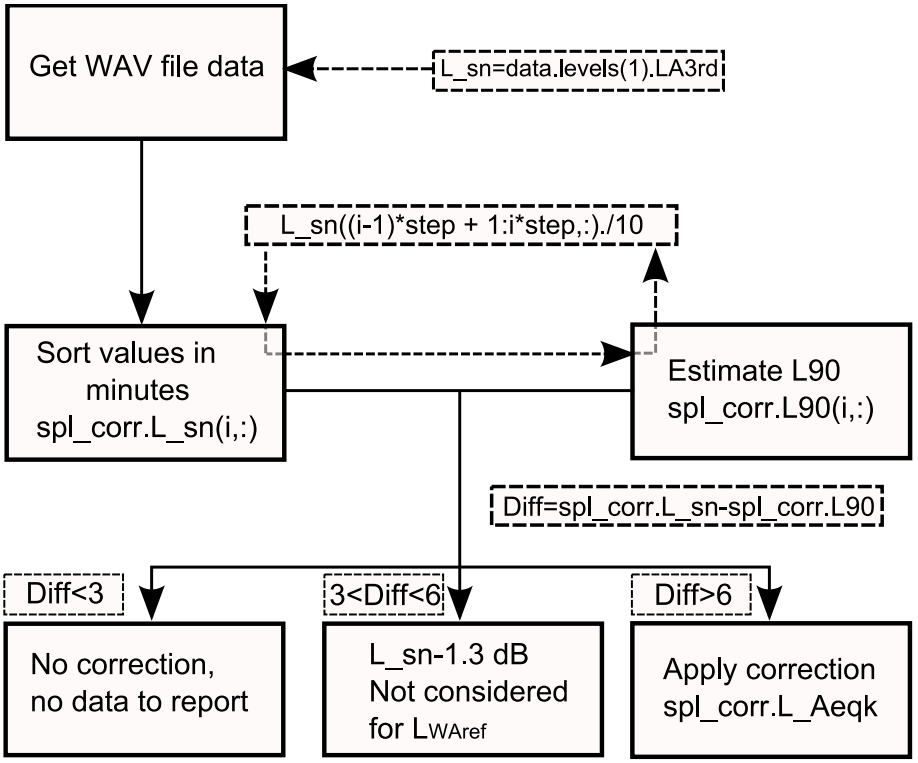

Figure 5.12: Calculation of background noise corrected sound pressure level

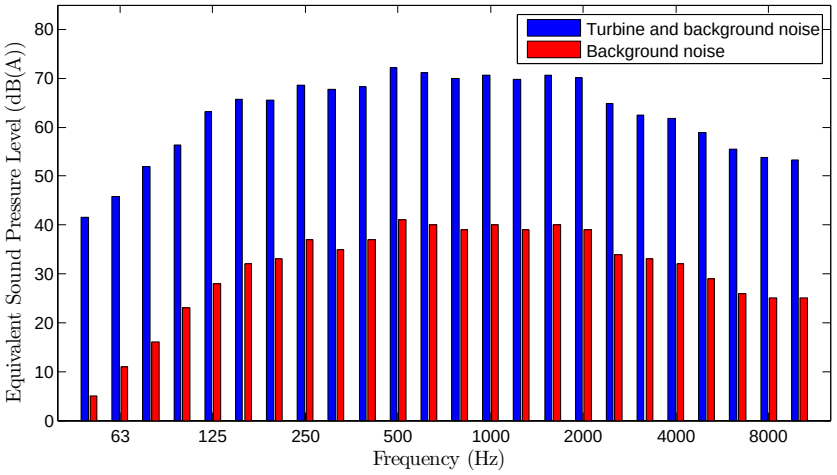

Figure 5.13: Equivalent continuous sound pressure level $L_{A,ref}$ vs. background noise L_{90}

5.3.3 Estimation of sound power of the turbine

The last step in this processing stage is to determine the apparent sound power level of the turbine for each of the measured wind speeds. *Figure 5.15* indicates how this is achieved. The values for each wind speed depend on previously computed background noise corrected sound pressure level in one octaves, the distance from the measurement microphone position to the turbine and hub height of the turbine. *Figure 5.16* shows the apparent sound power level of the turbine set of wind speed.

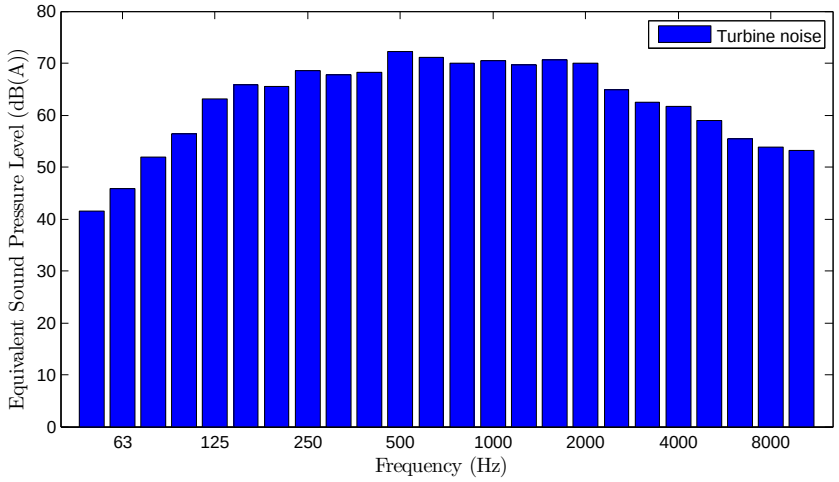

Figure 5.14: Background corrected sound pressure level, $L_{A,ref,k}$

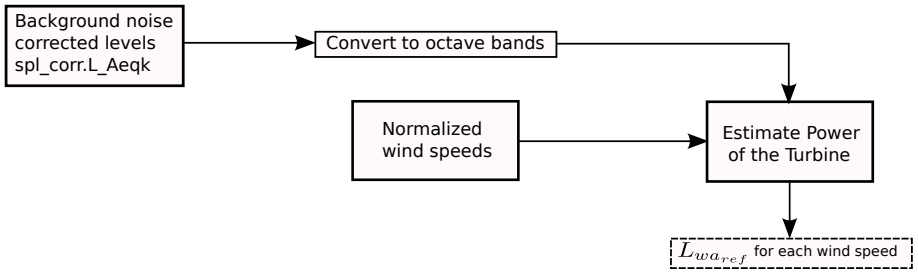

Figure 5.15: Post-processing stages

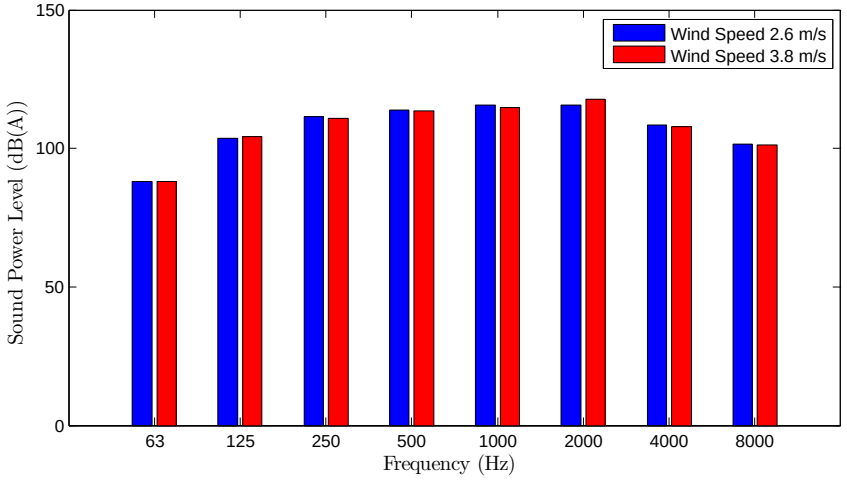

Figure 5.16: Sound power level of the turbine, $L_{A,ref,k}$, for wind speeds 2.5 m/s and 3.8 m/s

Part III

Data Treatment

The following part deals with the data treatment applied to the recorded data during the measurement session. First, based on results obtained in *Chapter 5* the sound pressure levels at a certain set of distances away from the wind turbine and the equivalent the day-evening-night levels, L_{DEN} at these exact positions are determined.
Subsequently, the recorded signals are inspected in order to find any significant noise artifacts. If this were the case, the signals would be rejected since they would not be adequate for playback purposes.
Lastly, a procedure to equalize the system "nanoLIAB + headphones" is given in order to perform an appropriate reproduction of the previously eligible signals.

6

Estimation of Significant Levels

This chapter presents a description of the levels that are presented to the final user of the implemented system. The levels calculated are the sound pressure level at a given distance ($SPL(dB)$), the equivalent A-weighted sound pressure level and the level day, evening, night $L_{DEN}(dB(A))$. For simplicity reasons, as the measurements did not take long enough to compute 12 hours of day, 8 of evening and 4 of night as stipulated in the standard ISO-1996 (see *Section 6.2*), the A-weighted sound pressure levels are assumed as the equivalent A-weighted sound pressure levels. For the calculation of these levels, considerations such as corrections for ground reflections, air absorption, type of terrain and frequency weighting have been taking into account. L_{DEN} is an important indicator when assessing road noise for instance and, although some research works [Chapter 4, page 43][Ped07] state that L_{DEN} is not important to assess wind turbine noise, it has been found interesting to calculate so the system can present an innovative feature in the case that L_{DEN} turns to be important in the evaluation of wind turbine noise

6.1 Sound pressure level and Equivalent A-weighted sound pressure level at a given distance

For the estimation of the sound pressure level emitted by the turbine at any desired distance, *Equations A.5, A.6* and *A.7* were considered. The sound pressure level at a given distance depends, apart from the distance, on factors such as the power level of the turbine, the air absorption and the type of terrain. It should be given in linear dB as these are the levels to be played by the nanoLIAB (see *Sections 8.1, 8.2* and *8.3*) and A-weighted, as this is the level required by the standard IEC 61400. The estimations of sound power radiated by the turbine are computed A-weighted. Therefore, they must be converted to linear values inverting the A-weight correction of each octave band [TRs] in order to calculate the linear sound power level.

The data used for the calculation of all levels is based on the field measurements carried out and used to determine the power level of the turbine. A correction of 6 dB was applied when determining the power level of the turbine in order to cancel the effect of placing a reflective surface (measuring plate) under the measuring microphone and suppose an ideal free field as defined in the

standard IEC 61400. Although the measurements were performed on the ground (see *Figure 6.1*), the above mentioned correction of 6 dB allows to consider equivalent the sound pressure level at the height of a person.

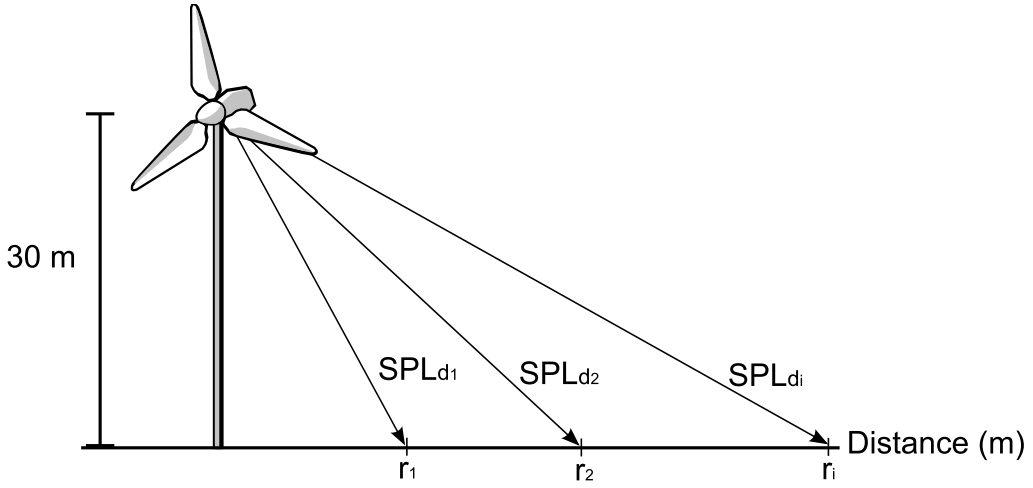

Figure 6.1: Sound pressure level calculation at any distance

Results of the estimation for different wind speeds are plotted in *Figures 6.2* and *6.3*. They show how the sound pressure level of the turbine would be perceived lower as both the wind speed and the distance increase. However, the short range of wind speeds captured during the measurement session does not allow to appreciate significant differences.

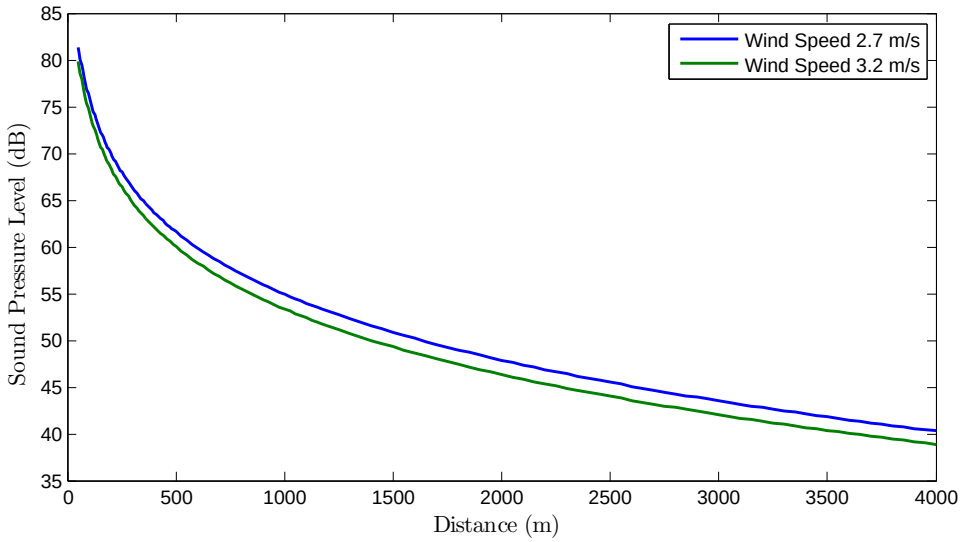

Figure 6.2: Sound pressure levels obtained for different wind speeds

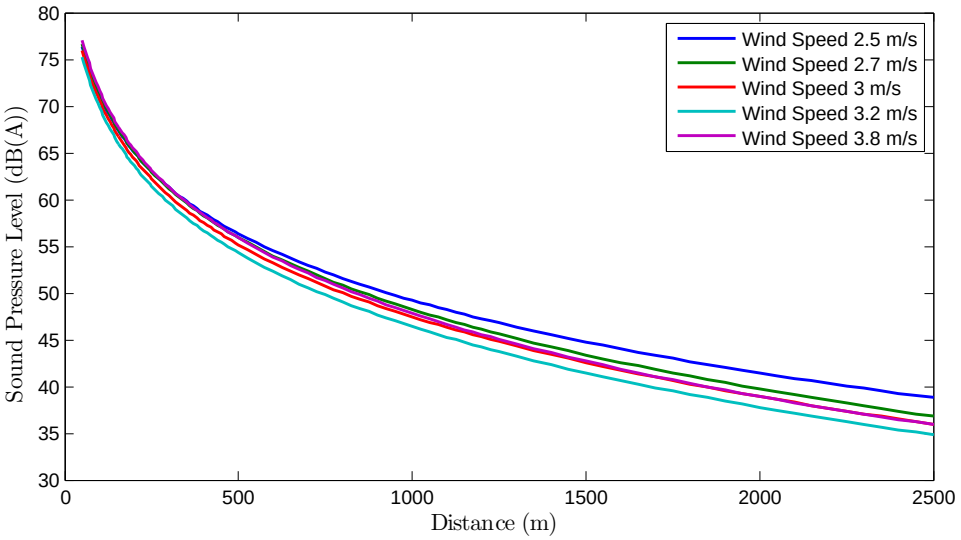

Figure 6.3: Equivalent A-weighted sound pressure level obtained for different wind speeds

6.2 Day-Evening-Night level estimation, L_{DEN}

For the estimation of the day-evening-night levels, L_{DEN} it is necessary to know the sound pressure level at a height from the ground of 4 meters according to the standard ISO-1996 *Description, measurement and assessment of environmental noise* [fS03], mentioned in the European directive EC/2002/49 *Assessment and management of environmental noise*[otEC02b]. During the measurement session these levels could not be measured so they ought to be computed from the data acquired on the ground. On *Section 6.1* it was shown how the equivalent A-weighted sound pressure level was estimated for any distance. Thus, from these aforementioned levels, the equivalent A-weighted sound pressure levels can be computed at any height. To calculate the sound pressure level at height of 4 meters, the following considerations must be taken into account.

- The sound pressure level measured at the ground presents a 6 dB increase due to the reflecting plate on the ground used to measure. This increase should be removed from the actual measured levels in order to obtain a fully free field estimation. This correction was already taken into account when computing the sound power level of the turbine from the ground measurements so it should not be done again.

- Assuming a single estimation of the sound pressure level from the ground measurements would provide the level on ideal free field conditions. L_{DEN} is an indicator of the equivalent sound pressure levels occurring in neighboring or industrial areas during a 24 hours interval in which penalties of 5 and 10 dB are added to the equivalent A-weighted sound pressure level during evening and night periods respectively. Therefore, it has been decided not to stick only to free field conditions. For a realistic situation, in which apart from the direct path followed by the sound from the source to the receiver a number of reflection occur as well, contributing to the determination of the total sound pressure level at an arbitrary distance. Since it is most interesting to present worst cases, an incoming in-phase reflection from the ground is considered. Thus, the total level at 4 meters can be easily computed with the data available (see *Figure 6.4*).

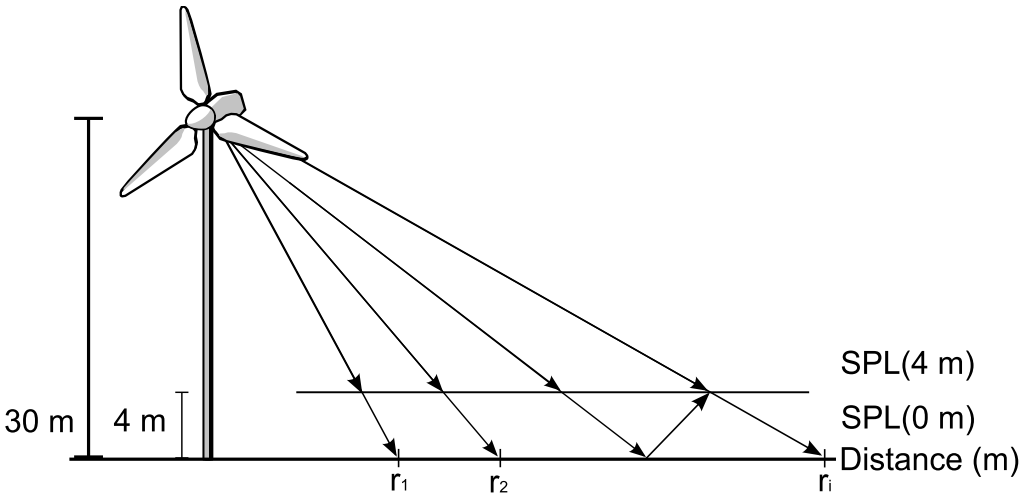

Figure 6.4: Sound pressure level calculation at 4 meters height for the determination of L_{DEN}

When the equivalent A-weighted sound pressure levels at the height of 4 meters have been obtained, the L_{DEN} can be calculated at any distance with the *Equation 6.1*, given on European directive EC/2002/49 as,

$$L_{\text{DEN}} = 10 \cdot \log\left(\frac{1}{24}\left(12 \cdot 10^{\frac{L_{\text{day}}}{10}} + 4 \cdot 10^{\frac{L_{\text{evening}\,+\,5}}{10}} + 8 \cdot 10^{\frac{L_{\text{night}\,+\,10}}{10}}\right)\right) \tag{6.1}$$

in which,

L_{day} is the A-weighted long-term average sound level as defined in ISO 1996-2: 1987, determined over all the day (12 hours) periods of a year

L_{evening} is the A-weighted long-term average sound level as defined in ISO 1996-2: 1987, determined over all the evening (4 hours) periods of a year

L_{night} is the A-weighted long-term average sound level as defined in ISO 1996-2: 1987, determined over all the night (8 hours) periods of a year

Estimations were done for different distances and wind speeds assuming an identical equivalent A-weighted sound pressure level during day, evening and night. Results reveal high L_{DEN} levels for distances up to two kilometers from the turbine (see *Figure 6.5*).

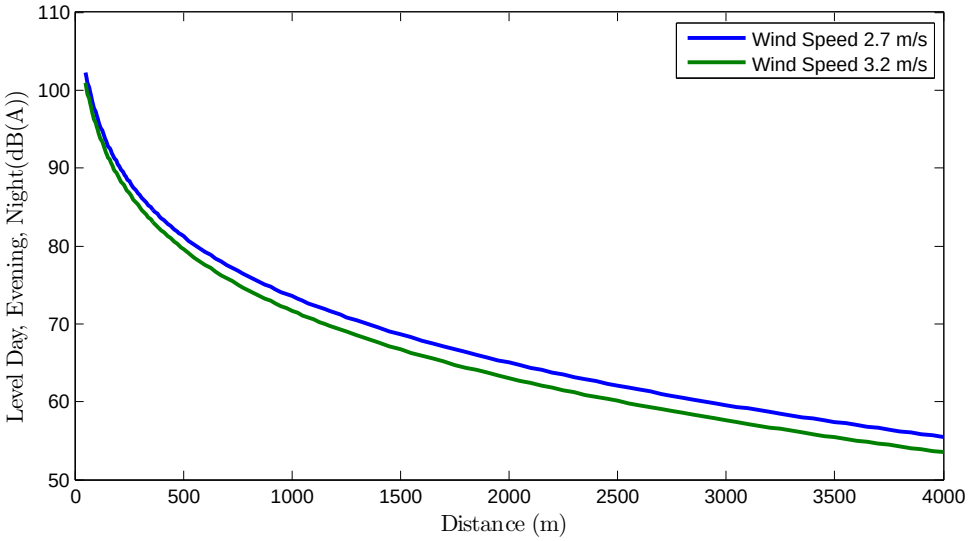

Figure 6.5: L_{DEN} estimations on different distances for a wind speed of 3 m/s

7 Preprocessing

This chapter intends to describe the listening inspection of the recordings needed in order to get the different signals from the original recordings that are going to be presented to the final user. Listening to the recorded signals is recommended by the standard IEC-61400 to detect audible artifacts in order to consider a possible rejection of the signal.

7.1 Inspection of the recordings

The aim of this section is to split the original recording of the wind turbine noise into different signals of 1 minute. As the wind data was collected every minute during the recordings for a time length of 19 minutes, this would lead us, obviously, to that amount of signals which are correlated with the measured wind speeds.

The original recordings were read in Matlab. As seen in *Section 5.2.2*, Harmonie records the data with a sampling frequency of 51200 Hz. However, the nanoLIAB only has the capability of reproducing signals at a sampling frequency of 44100 Hz and therefore re-sampling is needed. This is done by up-sampling the recorded data 44100 times and down-sample it 51200 times. Matlab allows to do this easily by using the function *resample*. Once the signals are properly generated, an individual inspection is performed on each of the signals. The purpose of this inspection is to find the presence of snaps or other artifacts as consequences of external noise sources or caused by an irregular performance of the wind turbine or external sources. It is important to remark that the turbine under assessment is more than 30 years old (see *B.1*. The measurement results do not vary so much since wind speeds present at the time of carrying out the measurements were very close. This drawback presents, however, the advantage that it is possible to discard signals that are considered to be "corrupted" by noise artifacts without loosing "perspective" since the recordings actually sound very similar. The results of this inspection can be observed in *Table 7.1*.

The results show a predominance of "dirty" signals which means that there is an audible presence of artifacts. These are artifacts are mainly identified with an irregular noise radiation from the wind turbine and should not be identified with the unique presence of external signals. In other words, these signals are in their majority suitable to be used for reproduction but all in all, wind speeds

Sound file	Wind speed (m/s)	Inspection result
minute 01	2.7	“Clean”
minute 02	2.9	“Corrupted”
minute 03	2.9	“Corrupted”
minute 04	2.6	“Corrupted”
minute 05	2.6	“Corrupted”
minute 06	3.8	“Corrupted”
minute 07	3.0	“Corrupted”
minute 08	3.0	“Corrupted”
minute 09	3.2	“Corrupted”
minute 10	3.2	“Clean”
minute 11	3.2	“Clean”
minute 12	3.1	“Clean”
minute 13	3.1	“Corrupted”
minute 14	3.1	“Clean”
minute 15	2.7	“Corrupted”
minute 16	2.7	“Corrupted”
minute 17	3.4	“Corrupted”
minute 18	3.4	“Corrupted”
minute 19	2.5	“Clean”

Table 7.1: Classification of the generated WAVE signals from the split of the recordings according to their wind speed and nature. “Clean” responds to an apparent absence of significant artifacts while “Corrupted” means that the signal is contaminated by some significant artifacts

are very similar and the actual artifacts are likely to be removed by repairing the turbine properly. For these reasons, it has been decided to remove them in order to get the signals as “clean” as possible. Thus, two signals considerably different (considering the wind conditions present during the measurement session) are selected for their reproduction, the ones correspondent to the first and tenth minutes.

8 System Equalization

In this chapter the process to equalize the system "nanoLIAB + headphones" is described. The recordings realized in the laboratory and their subsequent processing in Matlab are described. The purpose of this equalization process is to examine the behavior of the system when a sound is played back by the nanoLIAB. However, due to time limitations it was not possible to do the frequency study needed for the equalization. Instead, a simple calibration of the system has been performed. In other words, it is desired to investigate the linearity of each channel when applying different gains to either the WAVE signals reproduced or the mixer present on the system.

8.1 Gain modification, MLS recording

The sound emitted by a wind turbine will be presented to the final user via headphones. Therefore, it is important to know the exact sound pressure at the ears, given by the output of the headphones. By using a *MLS* signal generated in WAVE format it can be obtained the data directly as sound pressure. *MLS* signals are pseudo-random signals useful to determine the characteristics of a system. They enables to easily determine the impulse response of the system and check how linear it is [CVBG03]. Time limitations oblige to discard this option and determine the root mean square value of each *MLS* signal.

To check the linearity of the system, the nanoLIAB and the headphones are considered as a single block which output is recorded in two different experiments by modifying different gains. The first experiment consists of recording a WAVE file that presents a fixed level, modifying the mixer gain of the nanoLIAB. The second way inverts the process, for a fixed mixer gain of the Open Sound System, the WAVE file is recorded for a reproduction with different gains. For such purposes two scripts. *test_mixer_levels.c* and *test_wav_levels.c* were implemented.

Figure 8.1 shows the necessary setup in order to equalize and calibrate the system. As can be seen on the figure, the recordings of the signals were performed in a listening room and playbacks in an adjacent control room in order to minimize as much as possible the influence of any external source.

The Valdemar dummy head receives the output via headphones, connected to the nanoLIAB from

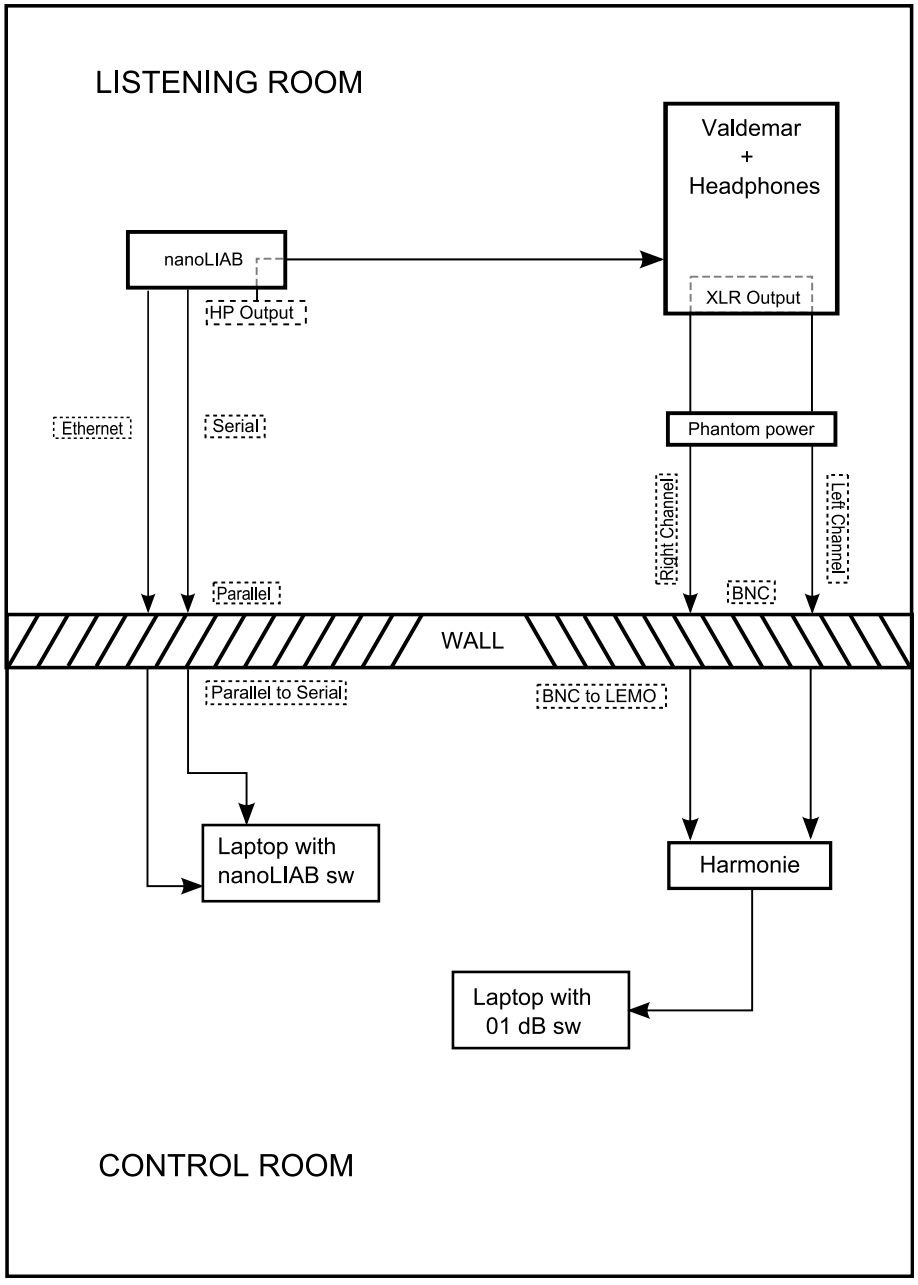

Figure 8.1: System equalization set-up

where the WAVE file is played. Also, the nanoLIAB is connected to a computer via Ethernet so it can be remotely controlled. The outputs of Valdemar are connected to the data acquisition platform, Harmonie. A phantom power feeds the microphones installed on Valdemar as they are condenser microphones. The Harmonie system collects the data as sound pressure (Pa. ref.20μPa.) by means of the software dBBati32 [dS] which allows to export the measured data in a text file, ready to be loaded and processed in Matlab. *Table 8.1* lists the equipment used for the equalization process.

Item	AAU No.
Dummy Head Valdemar	02150
DPA4037 omnidirectional condenser microphone (Valdemar left channel)	AAU56517
DPA4037 omnidirectional condenser microphone (Valdemar right channel)	AAU56516
Headphones Beyer Dynamic DT990	2036-14 AAU
nanoLIAB with software installed on it	2157-58
Laptop to control the nanoLIAB	
Laptop HP Omnibook 6000	47220 for dBBati32
01 dB Harmonie, data acquisition system	04124
01 dB Software and supplementary equipment	

Table 8.1: Equipment needed to perform the system equalization

8.2 Estimation of gain linearities

Prior to the recordings, it was necessary to know the maximum gain values of both the WAVE signal and the mixer in the nanoLIAB. The goal is to find a conclusion about the linearity of these changes (see *Section 8.1*). To determine the maximum gain possible to apply to the WAVE files it is known that a WAVE signal of 16 bits signed presents a maximum gain value of 32767 so the gains were applied in steps of 1000 from 0 to 32000. On the other hand, the mixer of the nanoLIAB presented a range of gains from 0 to 100 and signals were played back on steps of one. In total, 32 signals were played when varying the gain of the of the WAVE file and 100 when modifying the gains on the mixer.

Data recorded with Harmonie was exported to Matlab in order to study the changes. To analyze the data a small script was developed, *exeeq.m*. It looks for the beginning of each *MLS* recorded signal and extract it from the original data sequence in order to do not take into account the noise floor present in the signal. This is achieved by setting a threshold to ensure that what is being extracted is the *MLS* signal or at least a stable part of it, but nothing else. From each WAVE file, the root mean square (*RMS*) value was estimated, getting the sound pressure in pascals and its level in dB. *Figures 8.2, 8.3, 8.4* and *Figure 8.5* show the results of the gain variation for each channel on each case.

As can be seen in the plotted data, the behavior of the WAVE file when changing its gain in both channels is very linear when looking at the sound pressure. On the other hand, when modifying the gain in the mixer, the linearity is achieved in terms of sound pressure level. However, there is a difference in the pressure registered for both channels. The right channel outputs higher sound pressure whatever the gain is than the left channel. This can be seen easily in *Figures 8.6* and *8.7*.

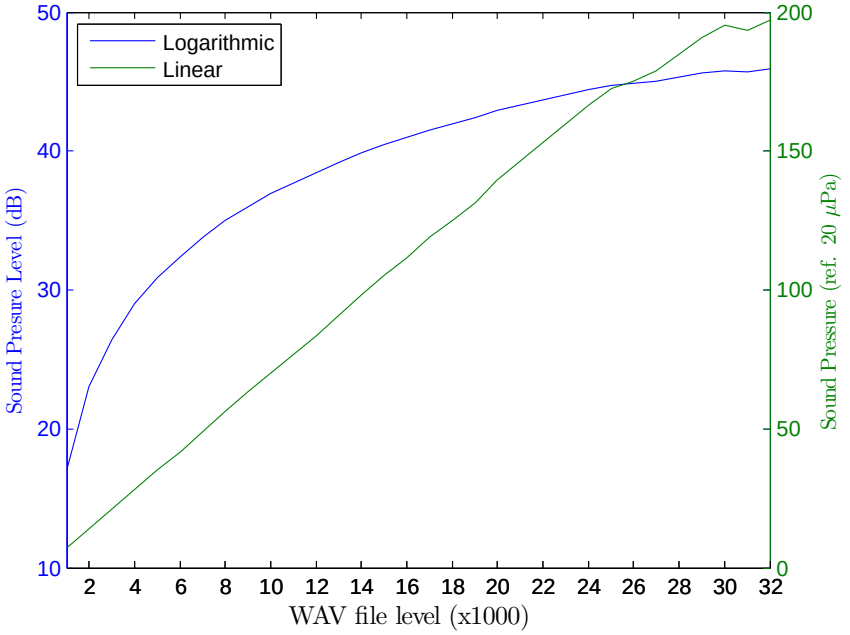

Figure 8.2: WAVE gain modification for left channel

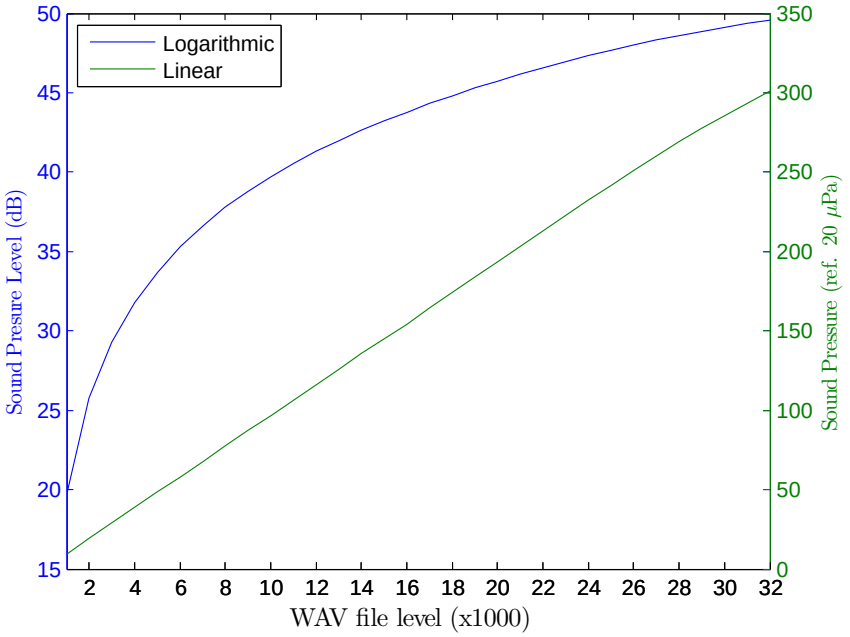

Figure 8.3: WAVE gain modification for right channel

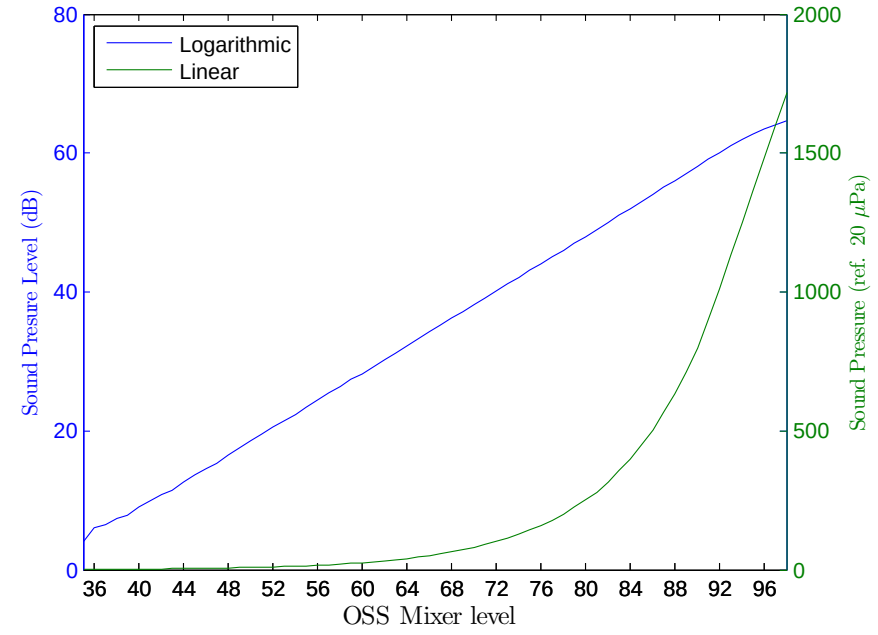

Figure 8.4: OSS mixer gain modification for left channel

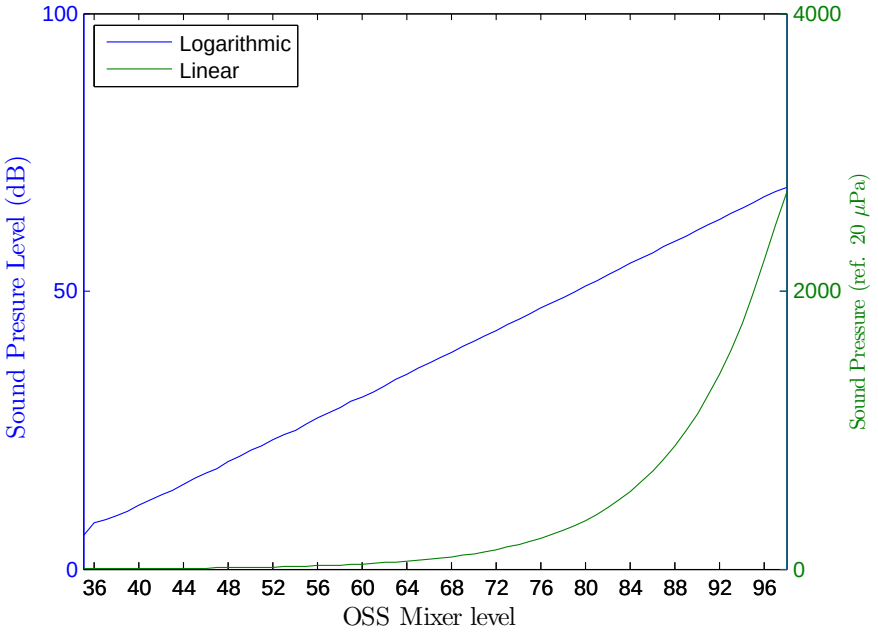

Figure 8.5: OSS mixer gain modification for right channel

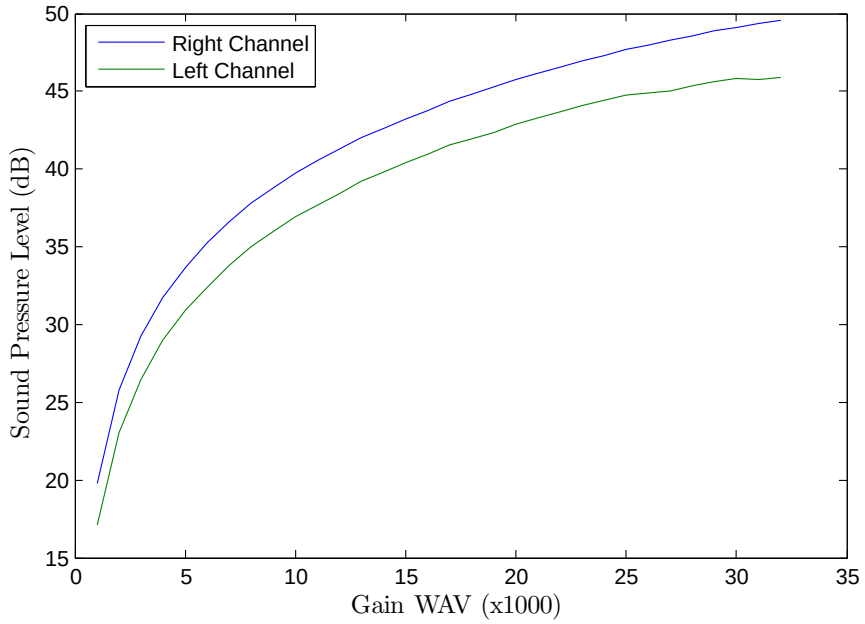

Figure 8.6: WAVE right vs. left channel comparison for constant gain changes

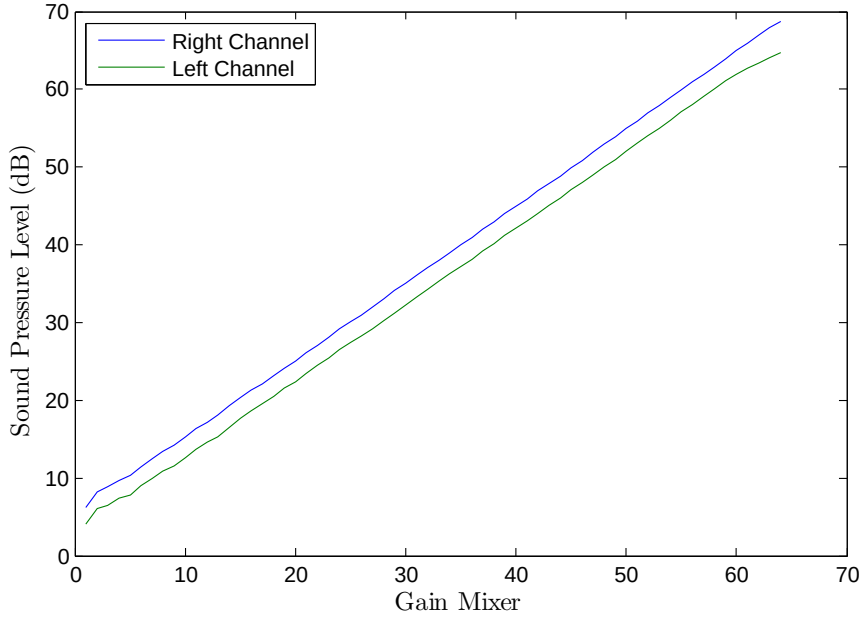

Figure 8.7: Mixer right vs. left channel comparison for constant gain changes

The level differences observed between the right and the left channel range from 2 to 4 dB depending on the gain. Two regions that covers the gains from 2000 to 20000 in the WAVE modification recording and from 45 to 94 in the Mixer modification recording resulted to be quite constant with mean gain differences of 2.93 dB and 2.79 dB respectively These are the gains that are considered from now on, discarding the rest. *Figures 8.8* and *8.9* show the difference between the two channels by subtracting one with the other.

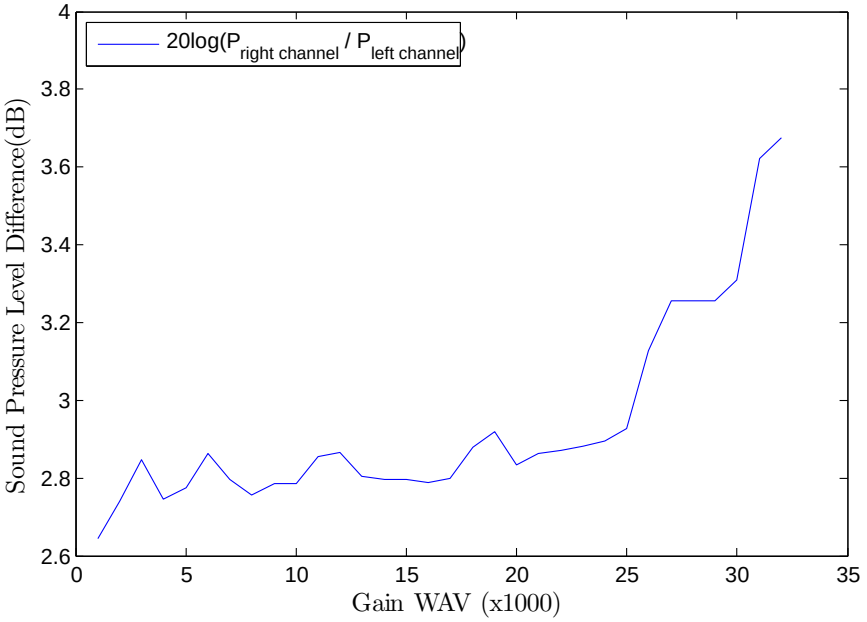

Figure 8.8: Right vs. left channel differences for constant gain changes in the WAVE file

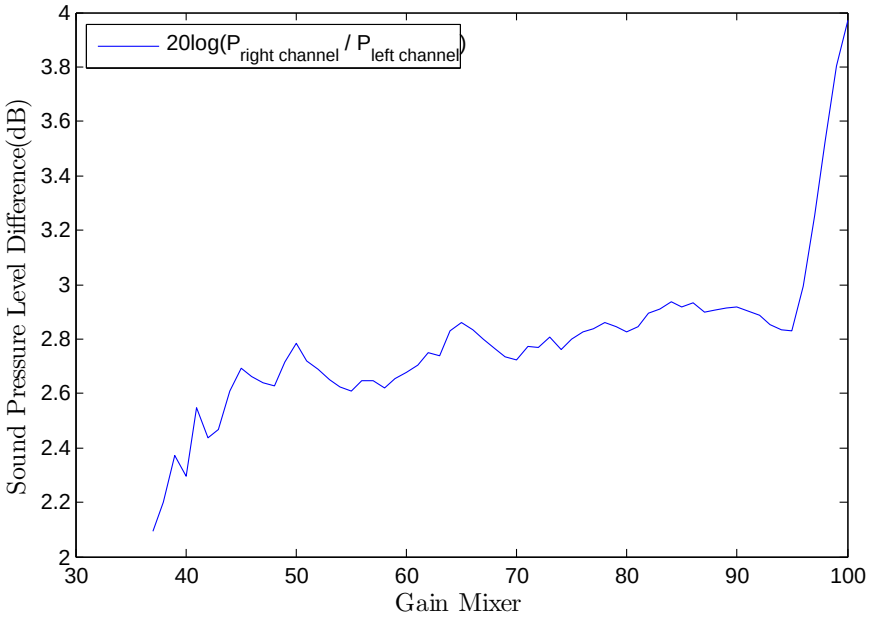

Figure 8.9: Right vs. left channel differences for constant gain changes in the mixer

Although it is easy to think that this is caused at the output of the system "nanoLIAB + headphones" this is not 100% clear since Harmonie sets the gain of each channel automatically prior to the recordings and it is possible that each channel was recorded with a different gain which may have an influence. It would be ideal to find out what causes this offset either by repetition of the recordings, by measurement of the impulse responses of the headphones or by measuring the output of each channel of the nanoLIAB with an oscilloscope. However, due to the limited time

available, it has been assumed that this is caused by the system "nanoLIAB + headphones". These gain differences imply a mandatory channel equalization.

8.3 Decisions based obtained results

In the previous section it could be observed how gain linearities were estimated based on the results obtained from the realization of two different experiments with the gain of the recorded WAVE file. It was kept constant while the gain of the mixer was changed and viceversa. The experiments were limited to two due to time restrictions. In the first one, the level of the WAVE file is fixed at a value of 16384 out of 32767. In the other one, the gain of the mixer is kept constant at a value of 75 out of 100. Ideally, it would have been adequate to realize different experiments of the same nature keeping fixed gains higher than the already existing. If this would have been the case, it could have been possible to ascertain the maximum level in both experiments at which linearity still remains achieving a better decision about which gain should be fixed and which gain should be kept variable could have been made. The reason of finding the highest linear value is that it would allow to reproduce a higher level of the wind turbine noise. In other words, the higher the sound can be played back, the smaller the distance to the turbine that can be reproduced. However, time limitations obliged to assume the obtained results as valid. A linear region has been determined for a WAVE signal level of 50% of its maximum value and a variable mixer gain within steps of one from 0 to 100 and for a mixer level of 75 and modifying WAVE level steps of 1000 from 0 to +32000. Another important aspect to take into account are the gain differences observed for each channel. This obliges to perform a channel equalization. An appropriate equalization, correcting the existing differences on each frequency band for both channels could not be achieved due to time constraints. Thus, it was decided to look instead at the common linear region of both channels and take their mean values to estimate a level difference between them. The resulting value (-2.79 dB) was considered as a correction factor to be applied on the right channel (the one with highest level) in order to have similar levels on both channels (see *Figure 8.10*).

The original signals were recorded as mono so they had to be converted to stereo and the above gain correction on the right channel applied. Furthermore, to adapt the noise files to the same correction as done with the *MLS* signals a new equivalent gain for the mixer must be found so that,

$$gain_{mixer} = 10^{-2.79/20} \left(\frac{rms_{mls}}{rms_{wav}} \right) \tag{8.1}$$

The gains obtained have to be corrected by a factor of one fourth to avoid clipping of the WAVE file. This results in an attenuation of 11.9 dB in the mixer to keep the same level that will have to be subtracted on each channel of the files (see *Figures 8.11* and *8.12*).

When both channels were considered to have the same levels for each gain change of the mixer a new linear region is then estimated for the corrected WAVE files with turbine noise recordings (see *Figure 8.13*). The maximum and minimum values of the linear region define the level limits available for a sound pressure level reproduction at different distances and the output that could be achieved on the nanoLIAB for every gain. *Table 8.2* details the sound pressure levels within the range of linearity available according to the previous estimation. All relevant data likely to be displayed, and the gain value or the mixer are included.

The data in the table is intended to show a typical set of data that would be sent to the nanoLIAB in order to play and represent in the user interface all information regarding the wind turbine noise.

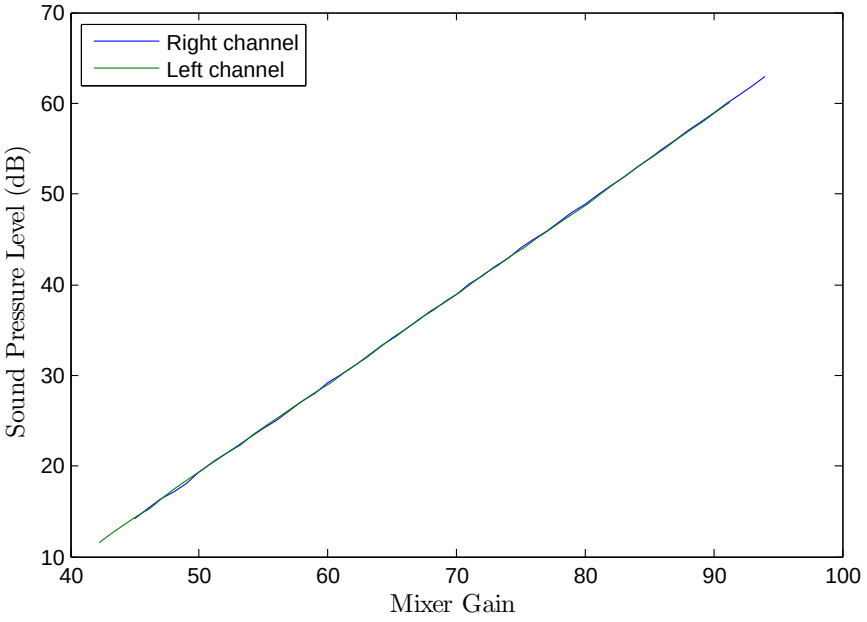

Figure 8.10: Right channel vs. left channel comparison after adding the mean linear region difference to the left channel

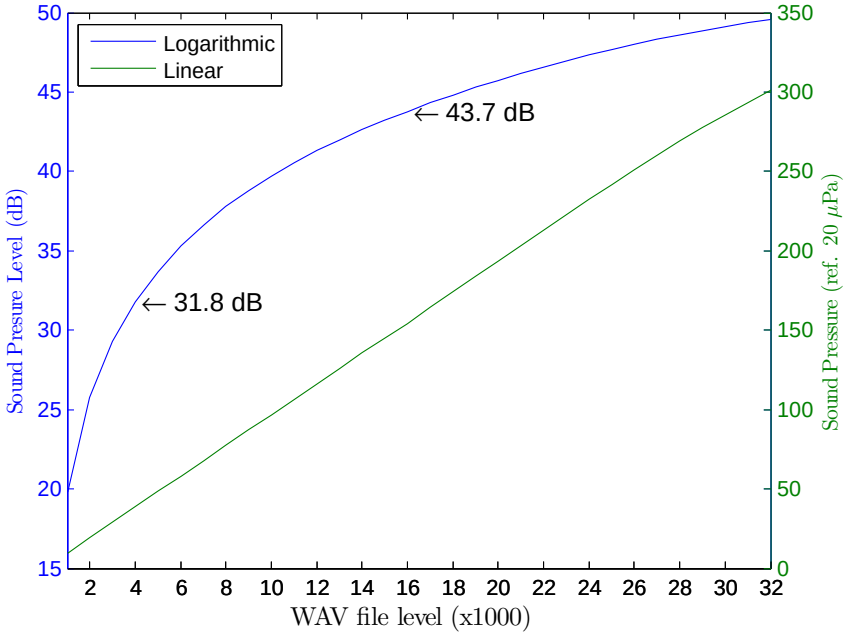

Figure 8.11: Correction to be applied for wind noise WAVE on right channel

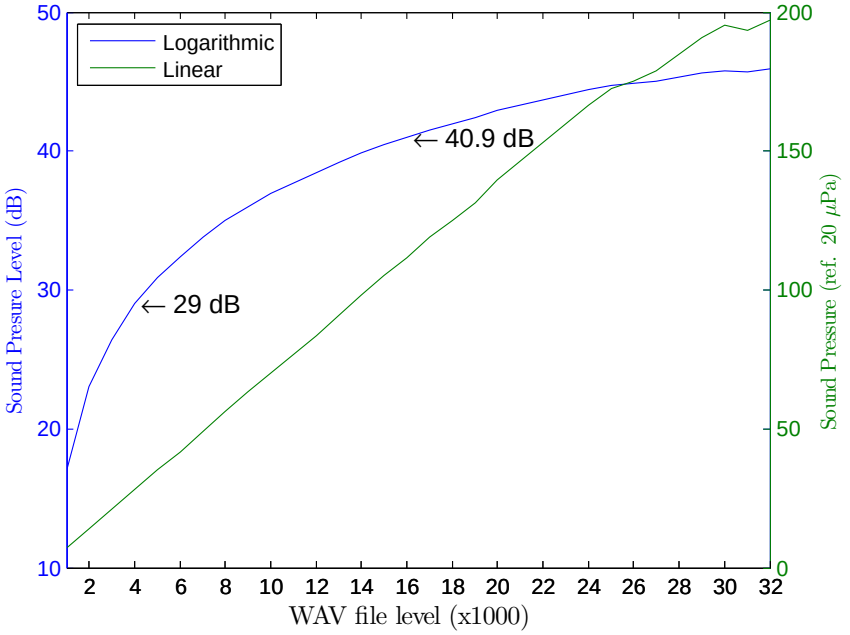

Figure 8.12: Correction to be applied for wind noise WAVE on left channel

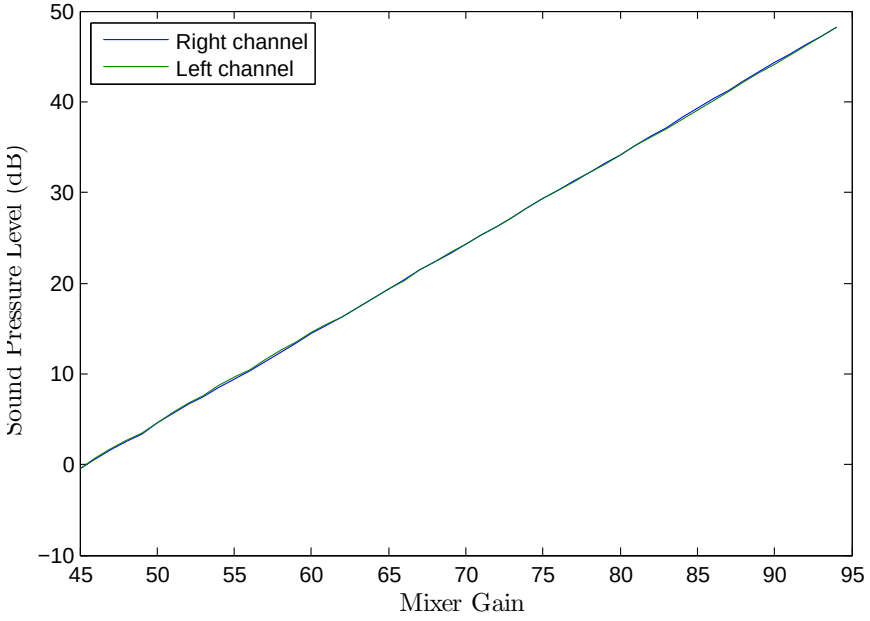

Figure 8.13: Correction to be applied for wind noise WAVE on left channel

Wind speeds (m/s)				
2.7				
3.2				
Sound power of the turbine (dB(A))				
$L_{WArefw=2.7m/s}$	101.77			
$L_{WArefw=3.2m/s}$	100.82			
Distance (m)	1950	2350	2850	3400
Sound pressure levels (dB)				
$SPL_{w=2.7m/s}$	48.16	46.18	44.1	42.14
$SPL_{w=3.2m/s}$	46.62	44.67	42.61	40.7
Equivalent A-weighted sound pressure level (dB(A))				
$L_{Aeqw=2.7m/s}$	40.1	37.7	35	32.8
$L_{Aeqw=3.2m/s}$	38.1	35.7	33	30.6
Day-Evening-Night level L_{DEN} (dB(A))				
$L_{DENw=2.7m/s}$	65.34	62.9	60.3	57.8
$L_{DENw=3.2m/s}$	63.36	60.9	58.3	55.8
WAVE files				
wind_min_1_st.wav				
wind_min_10_st.wav				
Mixer Gain	94	92	90	88

Table 8.2: Data needed for the presentation of the system

Part IV

System Design and Implementation

The design of the Noise Presentation System is divided into five parts.
The first chapter concerns the overall design of the system and gives an overview of the parts in which the system is divided.
Following the overall design, in *Chapter 10*, the platform upon which the rest of the system will be built is designed including the hardware of the nanoLIAB and the "base software distribution" that must be available for the rest of the system to function.
In order to play a WAVE file in the background with a predefined gain and change the input file or gain with minimal delay, when a new input is recieved, a piece of software has been programmed, which is documented in *Chapter 11*.
The User Interface is implemented as a web interface, and is documented in *Chapter 12*

9 Overall Design

The overall design of the system takes a starting point in the system requirements. From the requirements, a few considerations lead up to the overall system design, consisting of blocks described in the following chapters.

9.1 Requirements

In *Chapter 2* a number of basic system requirements were defined. The requirements were divided into system requirements and requirements related to the noise presentation, and are here presented again.

System requirements

These were the overall requirements of the system:

- The final system should be cost effective
- It should not be possible to tamper with the system
- The software should be possible to upgrade
- The system should be easy to use
- Option for linking multiple systems together to play the same sound simultaneously

Presentation requirements

For the sounds that can be played, it must be possible to see and/or select the following features:

- Law limits of the given noise (show and illustrate)

- $L_{A,eq(,T)}$ (select from predetermined values)
- L_{den} (show – select in "expert mode")
  - Day, evening and night levels might be relevant to show and edit – especially in other cases than wind turbine noise
  - For wind turbine noise, day, evening and night levels are assumed to be identical
- Distance in meters from noise source (select)
- Wind noise, optional: weather conditions of recording (show)
- Wind noise, optional: wind speed – choose 6 or 8 m/s
- Wind noise, optional: angle from wind direction (requires directional recordings)
- Road, railroad an airport noise: number and distribution of vehicles, trains or airplanes
  - Along with the day, evening and night levels, it might be possible to "synthesize" examples of the noise in the three cases and compare to the noise limits given by law

9.2 Design Considerations

From the requirements, the need to separate the user interface from the rest is definitely an important design parameter in order to ensure that the system cannot be tampered with. In *Chapter 3* it was decided to use a nanoLIAB board for implementing the Noise Presentation System. With the nanoLIAB, the separation is easily achieved by using a web interface and a pre-selected set of headphones as shown on *Figure 9.1*.

With the decision of using a nanoLIAB with a web interface for the control and presentation, it is relevant to consider what is required for this setup to work.

First of all, a way of playing back a piece of audio is absolutely necessary. This can be achieved either by finding a program for playing back audio, or by making a program for it. Also, it must be possible to run this program via the web interface without blocking the application. In order to run the program via the web interface, some type of server side scripting is required, for instance CGI, ASP or PHP.

With these considerations, it seems reasonable to have a system consisting of the following three blocks.

- A "base system" consisting of all that is needed to be able to show the user interface and play audio:
  - An operating system (Linux)
  - A module (driver) for the sound device
  - A web server
  - A scripting engine (PHP selected from the previous experience of the authors)
- Noiseplay: A program for playing the sound, which must be able to:
  - Play a WAVE file given as argument

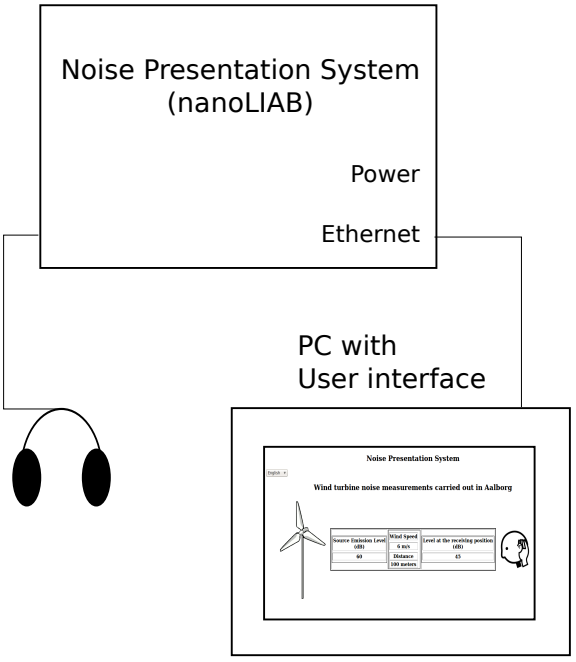

Figure 9.1: Overview of the noise presentation system as seen by the user

  - Play sounds in the background (return as quickly as possible)
  - Change the gain to a specified level
  - Play sounds in a loop infinitely or a predefined number of times
  - Change sound file or level with minimal delay (no need for playing the old file to the end)
  - Only allow one instance to be running at a time

- Web Interface to enable the user to control the system, which must be able to:
  - Show relevant information to the user
  - Allow the user to chose wind speed, source level and distance
  - Call Noiseplay with appropriate parameters

The interaction between the blocks and the user is illustrated in *Figure 9.2*, where the arrows indicate how information is passed through the blocks.

Another thing relevant to consider is the storage of the audio files, since only a small amount of non-volatile memory is available on the nanoLIAB itself (around 9 MB). Fortunately the device is equipped with a USB port, so a normal USB flash drive will be well suited for this task.

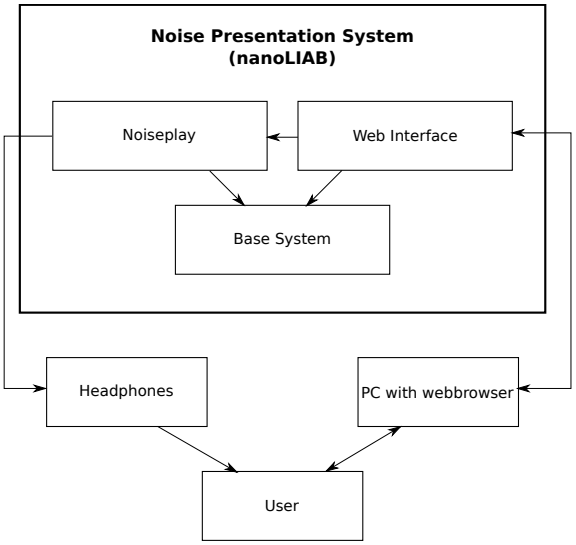

Figure 9.2: Block diagram of the different parts of the Noise Presentation System

9.3 Decisions

Based upon the considerations, the following decisions were taken.

- The main system will consist of a small closed box with a nanoLIAB inside
- A USB stick, with audio signals and information about them, will be connected inside the box, unavailable to the user
- A set of headphones will be connected inside the box, so they cannot be easily exchanged
- Plugs for power and ethernet will be available on the outside of the box
- The user will connect to the system via a PC with ethernet to a user interface programmed in PHP
- A program, noiseplay, on the nanoLIAB will correct the gain and play the sound. The program will be called through PHP
- The user will not be able to access the device internals, thus changing any part of the GUI or program

In order to meet these goals, it is necessary to design the three blocks mentioned above.

The base system is analyzed along with an examination of the hardware platform in *Chapter 10*.

Design and implementation of the embedded program to play the audio is described in *Chapter 11*.

And finally, in *Chapter 12*, design and implementation of the user interface in PHP is reviewed.

10 Platform

In *Chapter 3* it was decided to use a nanoLIAB board for implementing the Noise Presentation System, from an examination of the initial hardware requirements.

Hardware requirements

- A headphone output with full programmable volume control
- An interface for presenting and controlling the device
- Storage for the sound samples
- Optionally a method of communication between multiple devices
- A way to ensure that tampering is not possible either through hardware or software
- As cheap as possible

As a result of the overall system design, one additional requirement was added.

- The system must be able to process PHP

This chapter will examine the relevant details of the nanoLIAB more thoroughly with reference to the requirements and decisions will be taken, leading up to a design and implementation of the base system for the platform.

10.1 Description of the nanoLIAB

The name LIAB means “Linux in a Box” and the nanoLIAB is a small ARM-based microprocessor board with a number of external connections, running a Linux operating system when delivered.

Project relevant features in the system are:

- Small mini PC board (68×62 mm) w/ Linux already running
- 1w power consumption – allows for battery power if necessary, but it comes with a 7.5V DC adapter
- D/A converter and headphone amplifier to output sound in headphones
- No direct display support
- LAN interface allows an external to be used for display, control, communication – possibly via a web interface
- 16 MB FPROM for the base system and a limited amount of storage
- USB interface allows the use of USB storage easily
- AT91RM9200 ARM processor

10.1.1 The Supplied Distribution

Upon delivery, a CD with documentation of (almost) every part of the board, source code, cross compiler and scripts for generating disc images accompanied the nanoLIAB.

On the supplied disc image, an Apache web server was running and it was possible to mount a USB storage device when logging in.

When looking at the requirements of the system, there are two things not already available on the system, namely the nanosound module[1], which is needed to play audio and is available on the CD, and PHP which needs to be installed on the system.

Furthermore, the modules enabling the system to mount a USB flash drive are present, but not automatically loaded.

10.1.2 Memory Layout

Now, a more detailed review into the workings of the nanoLIAB is performed, using the default memory map of the FPROM, which is seen on *Figure 10.1*, as a starting point. On the figure it can be seen that there are five differently sized blocks allocated. The definition of the blocks as the five mtd device blocks are in the compiled Linux kernel, although obviously the boot loader must be in the first part.

The boot loader resides in the first 128 KB. When 3 dots are entered, it can be used to change the boot parameters, such as ip, and to upload uuencoded data[2] to the FPROM. If 3 dots are not entered, the boot loader will decompress the kernel and start the system.

The boot parameters, which are loaded upon boot, reside in the final 128 KB.

The Linux 2.6.16 kernel resides in the next 1536 KB after the boot loader. In the kernel, the 5 mtd blocks of the FPROM are defined, so they can be accessed with the proper tools.

[1] according to the documentation the nanosound module should have been available, on the jffs2 partition which is described later, but it was not

[2] uuencoding is a way of encoding any type of data to only use ASCII characters, intended to make it easier to transfer via a terminal emulator

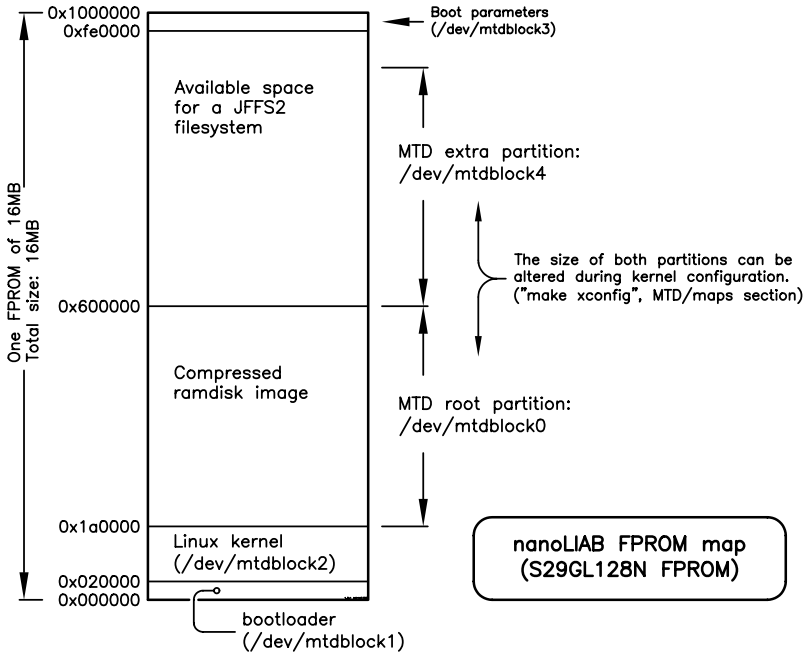

Figure 10.1: Default memory map of the nanoLIAB FPROM from the nanoLIAB user manual [LIA06]

A compressed root file system that takes up the next 4480 KB of the FPROM is uncompressed on boot and loaded into a part of the RAM, which is used as a file system.

The remaining block of 10112 KB is used as a part of the file system, that remains after power is turned off. This non-volatile memory is mounted at /jffs2 (Journaling Flash File System 2) on boot.

The pre-configured root image of the system is a small extract of the Debian sarge distribution, which is at the time of writing no longer officially supported. The binaries of the system are compiled with gcc-3.3.2 for the arm-softfloat-linux-gnu targets with the GNU C-library, glibc-2.3.2.

10.1.3 The Boot Procedure

Below is an overview of the boot procedure of the nanoLIAB with some side notes.

1. Hardware initialization
2. The boot loader checks if 3 dots are received via the serial interface
   - If 3 dots are received, the boot loader configuration is entered
   - If not, or when configuration is exited, boot is continued
3. The kernel is decompressed by the boot loader and loaded into the DRAM, starting at address 0x20008000
4. Final hardware setup, and copy kernel parameters to DRAM before the kernel takes over
5. Kernel decompresses the disc image to a RAM-disc

6. Boot is continued as defined in the settings of the RAM-disc (the init-scripts in /etc/rc.d/)

Take note that any data saved on the RAM-disc (root file system) will not remain after power off or reset. If one wish to change the contents of this system, it is therefore necessary to modify the data of the disc image residing in the FPROM. On the LIAB CD there are scripts available for generating a new such image, which can then be uploaded to the nanoLIAB.

10.2 Decisions

When comparing the requirements and the delivered system, it is noted that some changes to the root file system are necessary for the final system to work as intended directly on boot.

- The nanosound module must be added and loaded on boot
- The modules necessary for mounting a USB flash drive must be loaded on boot, and if one is available, it must be mounted
- PHP must be added so that it works with the web server

While editing the contents of the root system anyway, a few extra things might be nice to change

- The passwords of the system (saved in /etc/shadow)
- Run a "startup script" on the /jffs2 that can for instance do some initialization for the web interface or other configurations that could be nice to do without rewriting the entire disc image

10.3 Design and Implementation of the base system

In this section, the setup of the base system will be described.

Unfortunately the supplied cross compiler did not work properly, due to some file or link missing, so it was necessary to recompile the cross compiler using a script which is described in *Appendix C.1*.

The tasks of setting up the base system in the way it is described below and making it into a disc image has been achieved by making a shell script, called "make_base.sh" which is described in *Appendix C.2*.

10.3.1 Obtaining PHP for ARM

First of all, it was necessary to get a version of PHP, compiled for the ARM architecture, possibly in the form of a Debian package. Looking at the available Debian packages, they were all too large for the small size available for the nanoLIAB distribution. It was therefore decided to compile PHP with as few extensions and features as possible, and saving some space by using a smaller and more simple web server than the Apache 2 server originally on the system, THTTPD. The different Apache versions did not allow to compile PHP from the nanoLiab. Besides, a high amount of memory was occupied by Apache.

By following a guide found on the website of Mission Technologies[3], and adapting it into the *make_base* script, the THTTPD 2.21b web server was compiled with PHP 5.3.2 support into a 2.5 MB binary.

10.3.2 Making the Image

The next step after having compiled the web server with PHP support is to generate the image, which will be transferred to the nanoLIAB.

The original disc image can be found on the nanoLIAB CD, where the full distribution including documentation is packaged in the file *nanoLIAB-liab6I-ARM-may-2007.tar.bz2*. Extracting the file, which must be done with root privileges, as it contains device nodes, that must be present to generate the disc image.

The folder "software/liabdisc/libc6" contains the original root file system, which can be changed and compressed into a disc image with the script "mkfpromimage".

The nanosound module along with the device nodes /dev/dsp and /dev/mixer and some example program is in the compressed file "NanoSoundVer10.tgz" in the folder "software/liabaux". To install the module, the file should be extracted into the root file system (the libc6 folder). To start the module so sound can actually be output, the following command must be run on boot.

```
modprobe nanosound
```

By appending the command to the Linux initialization script, "/etc/rc.d/rc.M", the module will be started once the system is booted.

As mentioned earlier, the modules for mounting a USB flash drive should also be started on boot, and if a drive is attached, it should be mounted. This can be accomplished by editing the initialization script to include the following lines:

```
modprobe usb-storage &&
modprobe sd_mod &&
sleep 5 && # wait for the modules to detect if a USB disc is available
mount /dev/sda1 /mnt # try to mount it - gives an error to the boot screen, that can be
    ignored
```

Another change needed in the initialization script is to start the THTTPD daemon instead of the Apache daemon.

The last change to the script then, is to run a script from the non-volatile part of the memory, so the initialization can be modified without creating and loading an entire new disc image. It has been chosen to name the script "noise_setup.sh" and it is run by appending the following lines to the initialization script:

```
if [ -f /jffs2/noise_setup.sh ] ; then # run the noise_setup script if it exits
    /jffs2/noise_setup.sh ;
fi
```

It was also decided to change the password of the system, which is done by changing the password hashes on the "/etc/shadow" file, which looks like this:

```
root:$1$H6szsBbF$RXYb5ArDRJAgJn4zMvU981:11499:0:99999:7:-1:-1:134539268
nobody:*:12338:0:99999:7:::
apache:!!:12338:0:99999:7:::
liab:$1$r.cl8r5z$iboQ7ODryJ/M9uDa2Z.xB1:11499:0:99999:7:-1:-1:134540308
```

[3] http://www.missiontech.co.nz/index.php?page=read-an-article&articleId=102

The part to change in this file is the text between the first and second colon, for instance `$1$H6szsBbF$RXYb5ArDRJAgJn4zMvU981` to change the root password. The new hash can be generated by running the following commands, where $LIAB_PASS is a variable containing the password.

```
echo "$LIAB_PASS" | openssl passwd -1 -stdin
```

This will generate a new hash string, which will be different everytime.

As noted above, the maximum size of the compressed disc image is 4480 KB, which is not much when the THTTPD binary takes up 2540 KB alone. In order to preserve as much space as possible, an evaluation took place regarding which programs and files were necessary, and some files were stripped for comments. The "make_base" script was also made to compare the size of the generated disc image to the maximum size in bytes and report an error if it is too large. A different approach would have been necessary to recompile the kernel and redefine block positions. However, this was deemed unnecessary, and would require more work. Besides, it is desired to keep the space available for the web interface and prerequisites.

Finally, when everything in the root file system is as desired, the image is created and uuencoded so it starts at the correct start address of the FPROM.

10.3.3 Uploading the Image and Changing Boot Parameters

Once a disc image is successfully created, it needs to be uploaded to the system by following this procedure:

1. save the image as `d` in the folder from which it is intended to start the terminal emulator
2. Connect the nanoLiab to the com-port (before applying power) and connect with
   `cu -l /dev/ttyUSB0 -s 115200`
3. Hold `[.]` when plugging in the power to go into the boot loader and be able to upload the image to the nanoLiab. A screen like this will be loaded:

   ```
   ------------------o LIAB Bootloader o-----------------

   Release: 1.0, August 28, 2008 at 17:56 by midi
   Copyright LIAB ApS.
   The boot loader will now search for a compressed kernel
   and filesystem and try to boot up a Linux system.
   IF YOU WANT TO GET INTO THE BOOT LOADER, YOU MUST SEND
   3 DOTS WITHIN THE NEXT 5 SECONDS: *
   LIAB boot loader, 'h' for help
   Boot>
   ```
4. type f and [return] to enter the FPROM menu

   ```
   Boot>f
   Entering FPROM utility submenu, 'h' for help
   Memory configuration of LIAB
   AMD/Spansion, S29GL128N flash at addresses 0x00000000 to 0x00ffffff
   ```

```
FPROM>
```

5. Type `e 1A0000,5fffff` to erase a sufficient amount of the FPROM to put the image

```
PROM>e 1A0000,5fffff
Start address ...........: 0x1A0000
End address .............: 0x5fffff
Block size (Kbytes) .....:     4480
Do you want to erase? [y/n]>
```

6. Download the image to the nanoLiab: This is a bit tricky! Type l and return followed quickly by `~>d` and return again – NOTE! you only have around 1 second to do this!
   (to make it a little easier, copy `~>d` to your clipboard and type `l` and then quickly `<ret>, <shift>+<insert>, <ret>` instead)

```
FPROM>l
~>d
1 2 3 4 5 6 7 8 9 10 11 .... <-- indication of download
....   ....   ....   ....
72790 72791 72792 72793 72794
[file transfer complete]
[connected]
No errors during reception of uuencoded data
Start Address ...........:     1a0000
End Address .............:     4bfb2e
Length ..................:    3275567
POSIX.1 CRC checksum ....: 3810579789
```

7. Type `q, <ret>, q, <ret>` to boot the new system

It might also be necessary to change the network settings of the device if they do not fit to the local network available, since DHCP is not available on the system. If that is the case, this is also done via the boot loader configuration menu, by pressing `p, a` and then following the guide, entering proper ip address, sub-net mask, gateway/route and DNS.

11 Embedded Software

This chapter concerns the design of the embedded software called “Noiseplay”, which was made in order to play a sound through the headphones of the system.

11.1 Requirements

In *Chapter 9*, the basic requirements for the Noiseplay program were laid out.

Noiseplay: A program for playing the sound, which must be able to:

- Play a WAVE file given as argument
- Play sounds in the background (return as quickly as possible)
- Change the gain to a specified level
- Play sounds in a loop infinitely or a predefined number of times
- Change sound file or level with minimal delay (no need for playing the old file to the end)
- Only allow one instance to be running at a time

11.2 Design Considerations

Looking at he requirements, the main task can be defined as follows.

1. Process input arguments
2. Test that the settings are proper and that the WAVE file is of a proper format

3. Start a daemon[1] process if there is no process running, or change the settings of a running instance

   The daemon tasks are:

   3.1 Setup audio (only first time)

   In a loop:

   3.2 Load audio

   3.3 Do processing (gain and possibly other)

   3.4 Play audio

4. Exit with success as soon as the daemon is successfully started, or new settings have been set

11.2.1 Making a Daemon

In order to avoid having the system hang while playing, it is necessary to make the program run in the background as a daemon. When making a daemon, there are a few things that should be considered.

According to [Pot08], the minimum requirements when making a daemon are

- The `fork()` call is used to create a separate process
- The `setsid()` call is used to detach the process from the parent (normally a shell)
- The file mask should be reset
- The current directory should be changed to something benign
- The standard files (`stdin`, `stdout` and `stderr`) need to be reopened

For the purpose of debugging, it will also be a good idea to make it possible via a command line switch like `--no-daemon` to run the program without going into daemon mode.

11.2.2 Avoiding Multiple Instances

A commonly used method for only allowing one instance of a program to be running on a Linux system, is to use a lock-file.

One way of utilizing this method is to first open a file with read and write access (or create it if necessary). The file `lockf()` function is then used to test whether the file is locked, and to lock it if it is not. If the file is locked, then the program can either be set to exit directly or take a different action, like for instance updating the settings of the other process in some way.

[1] A daemon is a program which is designed to run as a service in the background

11.2.3 Updating Settings

As mentioned in the requirements, it is necessary to be able to change either the current WAVE file or gain settings while a sound is already playing. To achieve this, there are different potential approaches.

One method could be to simply kill the old process before continuing. This method is very simple and will definitely work[2]. It might however introduce some delay in the playing of the audio, since everything has to be setup again or cause other unpredicted problems.

Another method could be to use the unix builtin signals to send a command to the running instance. Signals, can be sent with the `kill()` function, which by default sends the signal `SIGTERM` (die gracefully), but can send a number of other signals as well, including a couple of user defined signals. This method might seem very simple, however there is no easy way to send a variable containing the gain or a filename by using this method, so one would need to transfer the actual data in some other way, like writing a file, which could then be read when a signal is catched.

Writing the settings to a file, which is then read by the running process seems to be the easiest way to transfer the settings given by the arguments to a running process without killing it. In order to avoid reading a file, which is still being written or writing to a file while it's being read, it will be necessary to employ the file locking technique both when reading and writing the file.

A way to avoid an overhead by continuously reading a file while playing, it would make sense to simply test whether the file exists, which can be done with the `access()` function, and if it does, then read the settings and delete it.

11.2.4 Playing Audio

In order to play audio, it is necessary to setup the DAC, which is done by using the driver from LIAB called nanosound. The driver is an OSS V.3[3] module, which at the current time supports only 44100 Hz Stereo in the `AFMT_S16_LE` format [LIA07]. The `AFMT_S16_LE` format means that the data must be signed 16 bit, little endian format with the channels interleaved.

Programming to output sound via OSS requires the use of two device files, that need to be set up, before further use.

`/dev/mixer` is used to setup the volume of the driver in various input and output channels[4].

`/dev/dsp` is the sound device in use (normally just a symbolic link, but on the nanoLIAB a direct device node). The device must be set up to use the correct format, number of channels and samplerate in that order, before any sound can be output [TS00]. When one wish to output sound it is done by writing a char buffer to the device.

In the Open Sound System – Programmer's Guide [TS00] are examples of how to properly setup and play audio with OSS, and it has been used as a reference guide.

On the CD there are two example programs avilable that show how to set up the mixer and sound devices and get information about their capabilites, namely *get_mixer_info.c* and *get_sound_info.c* in the *Software/Noiselplay/test/* folder.

In order to read a WAVE file to output, it is necessary to know how to get information from

[2] On the attached CD, an example of killing a daemon before starting a new one can be found in the file *Software/Noiseplay/test/daemon_kill.c*

[3] OSS V.3: Open Sound System API

[4] With nanosound only the "master volume channel" `SOUND_MIXER_VOLUME` is available

such a file. The Waveform Audio File Format (WAVE) is defined in a specification by IBM and Microsoft [IM91]. An overview of the header and other relevant information is available at `http://www-mmsp.ece.mcgill.ca/Documents/AudioFormats/WAVE/WAVE.html`.

The file *Software/Noiseplay/test/play_sound.c* shows an example of how to combine the above information to setup the sound device and play a WAVE file.

11.3 User Guide

In order to properly design the Noiseplay program, a small user guide was made to define how the program should work from a users point of view in the style of a unix man page.

NAME

noiseplay – play WAVE audio on nanoLIAB in the background

SYNOPSIS

noiseplay inputfile [options]

DESCRIPTION

Introduction

The Noiseplay program is a program for playing a WAVE file on the nanoLIAB system in the background. When called with no arguments other than the required filename, the WAVE file will be played in an infinite loop. With the options it is possible to change the gain and define how many times the sound will play. It is also possible to let the program output debugging messages and avoid going into the background.

Only one process will be running at a time. If a new process is started, the new file name and options will be transferred to the running process, which will start using the new settings immediately afterwards.

Options

filename – name and path of the WAVE file to play (required)

[-g | --gain] gain – Change the gain of the file. the gain should be a floating point number

[-m | --mixer] level – Change the level of the hardware mixer. The level must be between 45 and 94 to stay in the linear range of the mixer gain.

[-p | --plays | -l | --loops] number of plays – Change how many times the file playback should be repeated. Default is infinitely (-1).

[-d | --debug | --no-daemon] – Debug flag. If set, debugging messages will be output.
A new process will remain in the foreground (not go in daemon mode).
A running (daemon) process will start to log messages to */var/log/noiseplay.log*.

Examples

```
noiseplay wind_200m.wav
```

Plays the file *wind_200m.wav* infinitely at the original level.

```
noiseplay wind_200m.wav -l 0
```

Can be used to stop the playback of a process running in the background

Output

The program will exit with

Succes: when everything works out.

Error: if no filename is given or the first argument is not a valid WAVE file

Warning: if an argument is not recognized

11.4 Design and Implementation

Taking the requirements, considerations and user guide into account, a flowchart of the intended program layout was made, which can be seen on *Figure 11.1*.

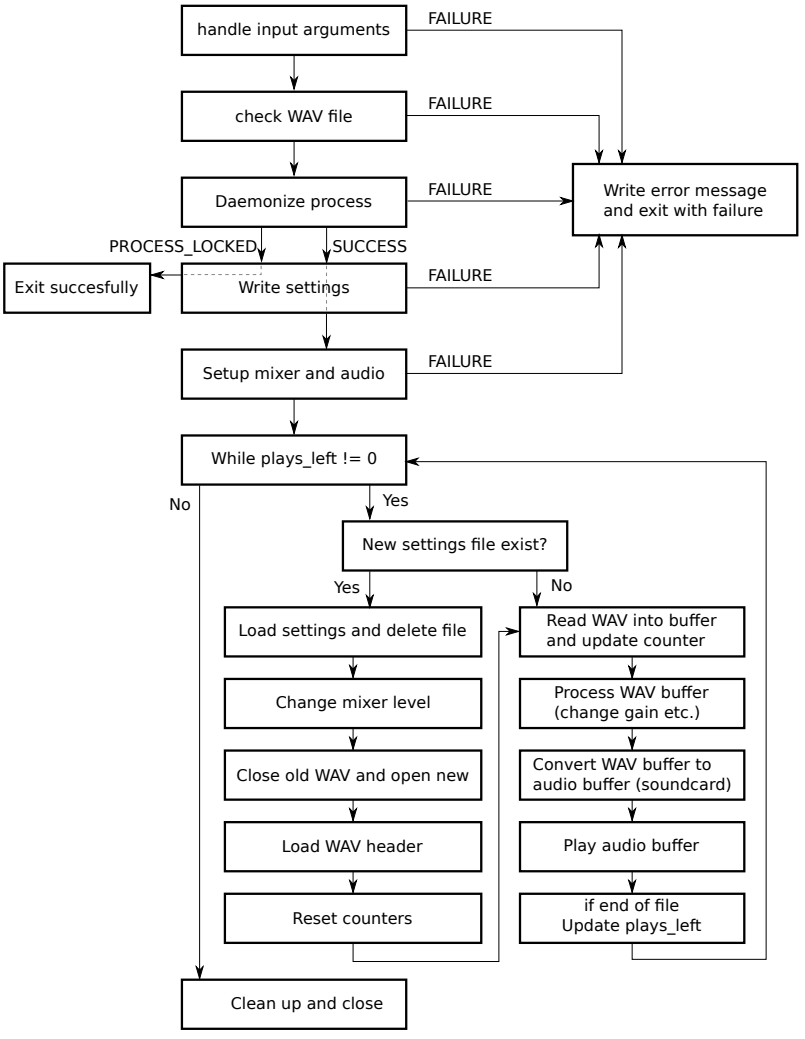

Figure 11.1: Flowchart of the main function of the Noiseplay program

The main idea is that only one instance of the program will be able to play sound at a time, and settings are transferred to a running instance by the use of a file containing the settings. In this way, it will be possible to use the same program with the same arguments regardless of whether a process is already running or not.

The program settings can be changed as fast as the time it takes to process and write one buffer

of audio. By setting the buffer length to for instance 1024 WAVE samples, with a sample rate of 44100 Hz, this means that the maximum latency of defining new settings will be around 23 milliseconds, which should be sufficiently low to meet the required need of changing the played file or settings "immediately" as perceived by the user.

In order to get a somewhat modular design, it was decided to split up the program into the following categories and related source files.

- Noiseplay – The main function and argument handling, which also includes every other part

  files: noiseplay.c, noiseplay.h

- Global settings and definitions

  files: global.h

- Daemon – functions for turning a process into a daemon

  files: daemon.c, daemon.h

- Sound – Soundcard interface functions: setup, buffering and playback

  files: sound.c, sound.h

- WAVE – functions for reading, checking and processing WAVE files

  files: wav_functions.c, wav_functions.h

- Settings – Functions for setting and changing the playback settings

  files: settings.c, settings.h

On *Figure 11.2* the relation between the functions and in which files they belong can be seen. A more detailed description of the functions in each group is available in *Appendix D*.

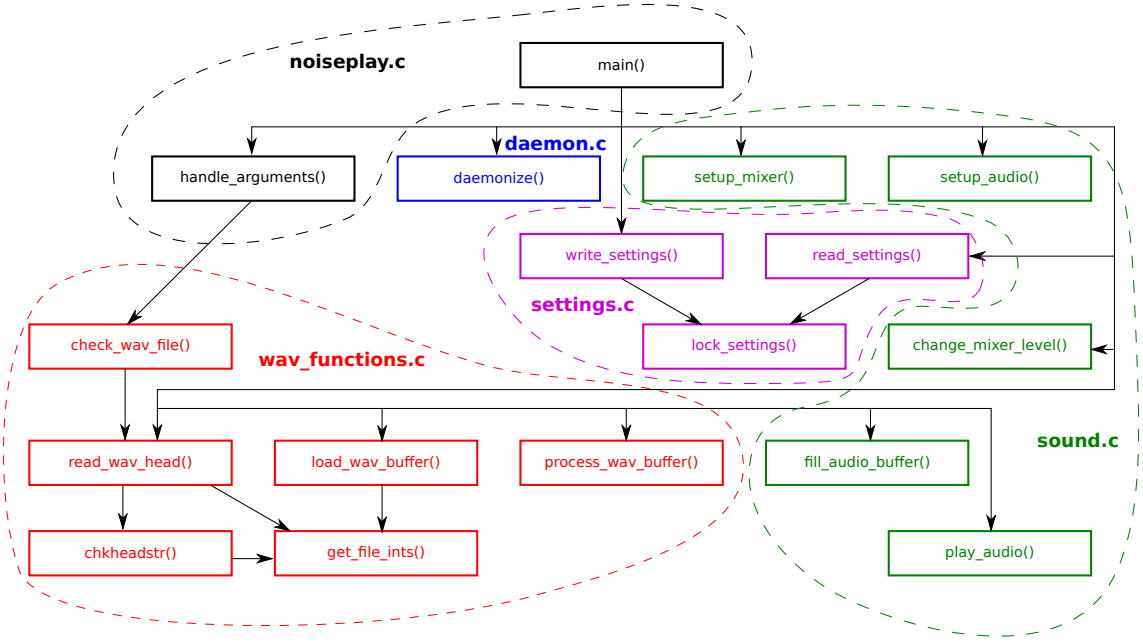

Figure 11.2: Flowchart of the various functions and how they are called.

12 User Interface

In order to enable a user to control the presentation system, a user interface has been constructed in PHP, which is made to be very modular, so it is easy to add other noise sources with each their own special requirements.

For now however, only the wind turbine noise module has been implemented.

12.1 Description

The user interface is inteded to control which levels to play as well as to complement the sound with relevant information by providing a visual tool that helps the user to better identify the sound being presented.

In order to work on an intarnational as wel as a local level, the interface hase been implemented with a system to cover multiple laguages with an easy selection and an easy way for translaters to translate.

In the interface, the sound power level of the turbine at the exact moment of the reproduction, the wind speed with which the sound was recorded and the sound pressure level at a variable receiver position are displayed while the sound plays in the background.

These features can be set by the presenter of the levels to fulfill the demands of the client so he/she can listen to different situations for a single wind turbine (see *Figure 12.1*).

As the worst situation is the most interesting to present, all distances have been calculated for a 0° orientation with respect to the turbine.

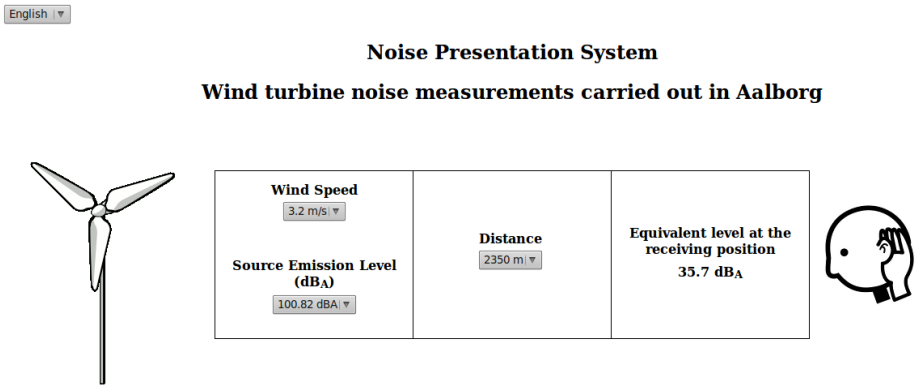

Figure 12.1: User interface of the wind turbine noise module

12.2 Requirements

In *Chapter 9*, the basic requirements for the requirements for the user interface were defined.

The Web Interface which enable the user to control the system, must be able to:

- Show relevant information to the user
- Allow the user to chose wind speed, source level and distance
- Call the Noiseplay daemon with appropriate parameters

The following requirements were set up in order to cover what the user interface must be able to do from a users point of view.

For the sounds that can be played, it must be possible to see and/or select the following features:

- Law limits of the given noise (show and illustrate)
- $L_{A,eq(,T)}$ (select from predetermined values)
- L_{den} (show – select in "expert mode")
  - Day, evening and night levels might be relevant to show and edit – especially in other cases than wind turbine noise
  - For wind turbine noise, day, evening and night levels are assumed to be identical
- Distance in meters from noise source (select)
- Wind noise, optional: weather conditions of recording (show)
- Wind noise, optional: wind speed – choose 6 or 8 m/s
- Wind noise, optional: angle from wind direction (requires directional recordings)
- Road, railroad an airport noise: number and distribution of vehicles, trains or airplanes
  - Along with the day, evening and night levels, it might be possible to "synthesize" examples of the noise in the three cases and compare to the noise limits given by law

12.3 Design Considerations

In order to make an application with a graphical interface, such as the one here made in PHP, there are a number of things that are relevant to consider first.

12.3.1 Features and Functionality

First of all, it was necessary to consider which features would be nice to have in the system, apart from the ones given in the requirements – both visible and internal.

Internally, it is alway nice for later maintenance to have the user interface seperated from the backend. This can be achieved in many ways, like for instance using a template system, which can be either very simple or rather complex. After searching for good solutions, the template engine "Smarty"[1] made a good impression regarding ease of use as well as available features and it was therefore chosen for the project.

Internationalization and localization is also a relevant feature to consider. Given that the project is aimed at the danish legislations and decision makers, it seems reaspnable that at least a danish version of the interface should be available. However, if successful, the project might also have relevance outside of Denmark, thus giving the need for an international version. For this purpose, the Smarty template engine showed to be of great value, with a class called smartyML [Rab04] which extends the Smarty class with multilanguange support.

Calling the noiseplay program should be done as a result of the user defined options and the predefined possible options. Because the noiseplay program is run directly on the nanoLIAB and not contained within the web server, it is necessary to make absolutely sure that the user cannot run arbitrary commands on the system for instance by setting some parameter in a way that escapes to the shell.

When looking at how the program should deal with input, it is desired to be able to reproduce certain settings simply by copying a link. This can be achieved by using GET requests to select the user defined settings, however this requires that utmost care is taken when handling the requests, so no arbitrary commands can be run. A good way to do this includes comparing all the set values to a list of allowed options.

12.3.2 Structure

The structure of the program as well as the folder layout should reflect the intent to keep the design modular and separate frontend from backend.

In order to make it easy to add different use cases of noise (or other sounds) to present, all which is not related to the general user interface shuld be made as "plug-in modules" for the topic-specific presentation and control. This is done by having the main PHP file search for valid modules and let the user decide between them.

In order to only have a single place to examine the input requests, all use of the program should be through one main file.

As mentioned in *Chapter 9*, the audio files and related information will be stored on a USB flash drive. An easy way of reading the data will be if it is stored as a variable in a PHP file, that can

[1] `http://www.smarty.net/`

then just be included.

12.4 Design and Implementation

Although it does not look like much, the design of the user interface has been focused og making it easy to expand in various directions.

- A new focus area of sound presentation can be added almost without modifying the rest of the program
- A new set of measurements and/or sound examples can be added simply by adding WAVE files and a PHP data file in the proper format to a USB disc (with a bit of work, it will be possible to autogenerate this file from Matlab while processing the data
- A new language can be added by copying a translation file, translating it and saving it in the lang folder

The layout of the files within the web directory has been laid out as follows:

- "docs" – documentation folder which includes coding guidelines followed for the implementation of a module, structure of each module and license of the software
- "images" – folder containing all common images of the user interface regardless the module that is being shown
- "include" – folder containin PHP files functions related to the general user interface
- "lang" – folder containing all translation files needed to present the interface in different languages
- "smarty" – folder containing all the common templates for Smarty template engine. For instance, this folder includes the templates containing html header and footer and the language selection button
- "modules" – A folder which contains the different plug-in presentation modules, such as wind turbine noise, road noise, airport noise or something else, each in their own folder. Due to time limitations and since this thesis focuses on a single noise source, only the wind turbine noise module has been implemented. A typical module would include the following,
  - the file module.php must be present and define the module and connect further files
  - "images" – a folder containing all relevant images intended to add visual aids which describe accurately the noise source (e.g. a picture of a wind turbine) enhancing the listening experience
  - "smarty" – folder with all template structure necessary for a specific module. For instance, the module of road noise may include a template in which the day-evening-night level L_{DEN} is presented but it would be not necessary to include it in the wind turbine noise module
  - a language folder with the specific vocabulary used when referring to a specific noise source in the available languages of the interface

The layout of the USB disc, which is automatically mounted on /mnt/ at boot time is so that there is a folder with the name of the module to which it fits. Within this folder is a file called “data.php” and a folder *Sound* with the sound samples. The file data.php, has no other function that to define a variable, $data, with all relevant information about the accompanying signals. The structure of data in the case of a wind turbine is as follows, taking an example with two wind speeds and four different distances:

```
1  $data=array(
2  "soundpath" =>"/mnt/wind_turbines/sounds", // string with the path of the WAVE files
3  "power"     =>array(
4          "ids"   =>array(101.77,100.82), // array of floating point representations of the
               radiated power (fitting the wind speeds)
5          "names" =>array("101.77 dBA","100.82 dBA") // array of strings related to the
               radiated power levels - the text that will be in the drop down boxes
6          ),
7  "distance"  =>array(
8          "ids"   =>array(1950, 2350, 2850,3400), // integer array with distances from the
               turbine in meters
9          "names" =>array("1950 m", "2350 m", "2850 m", "3400 m"), // related strings to
               show in the drop down box
10         ),
11 "wind"      =>array(
12         "ids"   =>array(2.7,3.2),// floting point array of wind speeds
13         "names" =>array("2.7 m/s","3.2 m/s"), // related strings
14         "speed" =>array( // main data part - subdata for each wind speed
15             "2.7"   =>array(
16                 "distance"  =>array(1950, 2350, 2850, 3400), // distances for this wind
                       speed
17                 "power"     =>array(101.77), // Radiated power for this wind speed
18                 "Alevel"    =>array(40.1, 37.7, 35, 32.8), // A weighted levels for this
                       wind speed at the given distances (Comparable to the law limits)
19                 "spl"       =>array(48.16, 46.18, 44.1, 42.14), // Levels to play,
                       without weighting
20                 "lden"      =>array(65.34, 62.9, 60.3, 57.8), // estimated day, evening,
                       night levels
21                 "files"     =>array("wind_min_1_st.wav","wind_min_1_st.wav","
                       wind_min_1_st.wav","wind_min_1_st.wav"), // files fitting the
                       different distances (in a later version, they would have been
                       filtered differently to simulate the air absorption)
22                 "gain"      =>array(94, 92, 90, 88) // the level which the mixer of the
                       nanoLIAB shouldbe set to, in order to play at the correct level.
23                 ),
24             "3.2"   =>array( // Data similar to the above...
25                 "distance"  =>array(1950, 2350, 2850, 3400),
26                 "power"     =>array(100.82),
27                 "Alevel"    =>array(38.1, 35.7, 33, 30.6),
28                 "spl"       =>array(46.62, 44.67, 42.61, 40.7),
29                 "lden"      =>array(63.36, 60.9, 58.3, 55.8),
30                 "files"     =>array("wind_min_10_st.wav","wind_min_10_st.wav","
                       wind_min_10_st.wav","wind_min_10_st.wav"),
31                 "gain"      =>array(94, 92, 90, 88)
32                 )
33         )
34     )
35 );
```

12.4.1 Dynamic Functions and Security

Just like loading a PHP file with data allows the data and WAVE files to be changed easily, it is possible to get the available languages and modules in a generic way.

This is done by getting a list of relevant files in the directory structure and test whether they are useable.

First of all, it is necessary to get information about the relevant files, for which the following two

functions have been implemented.

```
// get array of files without directories filtered by extension
function file_list($dir,$xt)
{
    $files=array(); // define the empty variable
    foreach(array_diff(scandir($dir),array('.','..')) as $file)
    {
        if( is_file($dir.'/'.$file) && ( ($xt)?preg_match("/$xt$/",$file):1) )
        {
            $files[]=$file;
        }
    }
       return $files;
}

// get array of directories
function dir_list($dir)
{
    $dirs=array();
    foreach( array_diff(scandir($dir),array('.','..')) as $file )
    {
        if( is_dir($dir.'/'.$file) ) // tells wether a file name is a directory
        {
            $dirs[]=$file;
        }
    }
    return $dirs;
}
```

For the case of getting a list of the available languages, the function below was implemented, which first gets a list of all the files in the *lang/* folder with the extension *.lang* and then extracts the translated name of the language from each file to use in the language selection.

```
// get language files in dir
function get_langs($dir)
{
    $langs=array();
    $lang_names=array();
    $files=file_list("$dir",'.lang'); // find files with .lang extensions
    if($files)
    {
        foreach($files as $file)
        {
            $thislang=explode('.',$file); // split filename into array at . (eg $file='en
               .lang' => $thislang= array('en','lang') )
            $langs[]=$thislang[0];

            // search translation file for the language name (html encoded)
            $lines=file("$dir/$file");
            foreach ($lines as $line)
            {
              $translation_pair = explode('=',$line);
              if(trim($translation_pair[0])=='LANG_NAME')
              {
                $lang_names[]=($translation_pair[1]);
                break;
              }
            }
        }
    }
    return array ($langs , $lang_names );
}
```

In order to set the language as the result of a user input, it is necessary to read the requested language from the GET parameters. The function below will attempt to read the language from _GET and compare it to the list of valid languages returned from the function above. If the request is valid, the function will return this language id, otherwise it will default to english.

```
// set language from url, if it exists - otherwise, default to english
function chose_lang($_GET,$valid_langs=array('en'))
{
    $lang=LANG_DEFAULT; // initialize to the default language
    if( isset($_GET['lang']) )
    {
        $_GET['lang']=rtrim($_GET['lang'],'/'); // remove trailing /'s to fix thttpd's
            tendency to put / after the url if not called with filename
        if( in_array(strtolower($_GET['lang']),$valid_langs) )
        {
            $lang=strtolower($_GET['lang']);
        }
    }
    return $lang;
}
```

Generically reading and selecting the modules can be done in a similar way to the selection of languages. The most relevant thing is, for security reasons, to always remember to initialize the variables and to remember to validate the input against the legal values.

In order to limit the possible places a security issue can hide, it was decided to only allow the index file to be called directly. This can be done by defining a constant in *index.php* which is then tested for in all the other files.

```
define('NOISE_PRESENTATION', true);

// security - do not allow direct access to this file
defined('NOISE\_PRESENTATION') || die("No direct access to this file");
```

12.4.2 Template System and Translation

It was decided to use the template system Smarty, which according to [OZ07] can best be described as a "Template/Presentation Framework", whose primary goal is to "facilitate the separation of application code from presentation", which is exactly what is needed.

In combination with the Smarty Multilanguage Support class, smartyML [Rab04], it was also possible to separate translation and presentation.

To setup the system for use, the following part of the initialization file is required:

```
define('LANG_DEFAULT','en',true); // set the default language

// set timezone
date_default_timezone_set('Europe/Copenhagen');

// initialize smarty template system and multilanguage support
// put full path to Smarty.class.php to use the functions
require_once '/usr/local/lib/php/Smarty/Smarty.class.php';
require_once 'include/smartyML.class.php';
$smarty = new SmartyML();

// setup directories for smarty
$smarty->template_dir = 'smarty/templates';
$smarty->compile_dir = 'smarty/templates_c';
$smarty->cache_dir = 'smarty/cache';
$smarty->config_dir = 'smarty/configs';
```

As can be seen, a number of directories are set up to be used by the smarty class. In order to use different template directories, as is wanted in the case of the modules, one simply has to create a new instance of the class, for instance with

```
$smarty_wind = new SmartyML();
```

To set up the system to use a specified language, the following is executed, where `$lang` is a string containing an abbreviation of the language to change to, for instance “en” which is the default.

```
$smarty->language->setCurrentLocale($lang);
$smarty->language->loadCurrentTranslationTable();
```

Normally, translation is done and requested in the template files, however if needed, translation of a specific key can be read into a variable like this

```
$lang_name=$smarty->language->getTranslation('LANG_NAME');
```

Assigning variables to the template and displaying the output is done in the following way

```
$smarty->assign('lang_ids',$valid_langs_main); // apply a vector of the id's, which will
   be converted into GET requests
$smarty->assign('lang_names',$valid_lang_names_main); // vector of strings, which will be
    the text in the dropdown meny
$smarty->assign('lang',$lang); // this is the currently selected id
$smarty->assign('remaining_gets',remaining_gets($_GET,'lang')); // append any further get
    requests like they are to the new page

// display separate parts of the page from different templates
$smarty->display('header.tpl'); // html header - right now only start tags and title,
   ending with the <body> statement

$smarty->display('lang.tpl');

$smarty->display('footer.tpl'); // footer - right now just closing the body & html tags
```

Where the contents of the templates, which are files in the template folder set up above, can be like the following

header.tpl

```
<html>
<head>
<title>##TITLE##</title>
</head>
<body>
```

lang.tpl

```
<form>
    <select name=lang ONCHANGE="location = '?lang=' + this.options[this.selectedIndex].
        value + '{$remaining_gets}';">
         {html_options values=$lang_ids output=$lang_names selected=$lang}
    </select>
</form>
```

footer.tpl

```
</body>
</html>
```

Notice how the header template contains the string `##TITLE##`, which is the format of a translation key. This means that the window title will be set to the title defined in the currently used language file, which is a simple text file, in which the text should be HTML encoded. For instance, the english translation file *lang/en.lang* contains the line

```
TITLE=Noise Presentation - Wind Turbines
```

And the danish translation file *lang/da.lang* contains the similar line

```
TITLE=St&oslash;jpr&aelig;sentationssystem - Vindm&oslash;ller
```

The result of the compiled templates above, seen by the user, will result in the dropdown selection menu seen on figure 12.2 where also the translated title can be seen.

Figure 12.2: Language selection menu and related window title.

12.4.3 Playing Audio

In order to play audio samples, the external Noiseplay program has to be called with all the proper dynamic arguments.

For obvious reasons, it would be a bad idea to allow the user input to be used for these parameters directly, as the user would then potentionally have full access to the entire system.

First of all the parameters will have to be read from the user and then converted to the parameters relevant to the noiseplay program.

In the wind_turbines module, this is implemented after using the GET parameters and the loaded `$data` array to read the selected wind speed, emitted power of the source and distance. In the `$data` array, the proper noiseplay parameters can be found and extracted by indexing the arrays with the parameters of the user interface.

```
$mxgain=$data["wind"]["speed"]["$wind_speed"]["gain"][array_search($distance,
    $distance_ids)];
$wavfile=$data["wind"]["speed"]["$wind_speed"]["files"][array_search($distance,
    $distance_ids)];
```

Finally, using the exec function, the noiseplay program can be called

```
exec("noiseplay {$data["soundpath"]}/$wavfile -m $mxgain 2>&1",$output,$return); //play
    audio
```

The `2>&1` part of the command is a method for redirecting stderr to stdout, because error messages cannot othewise be shown. If anything is not working properly, then the `$output` and `$return` variables can be examined to see the output and exit status of the program.

Part V

Conclusions

13

System Test

This chapter contains a brief description of the system test, that should be carried out. Due to time restrictions, no proper test was carried out on the system, however a very simple test shows that the system as such works.

13.1 Test design

In order to test the noise presentation system, Harmonie will be used to determine the levels of the output of the system, while different distances and wind speeds are selected.

The test of the system should be carried out in a manner similar to the attempts to equalize the system in *Chapter 8*.

The setup of the system test can be seen on *Figure 13.1*.

In order to avoid the influence of any external source, the measurements will be carried with the system, headphones and dummy head in a closed room in the acoustics laboratory and all remaining equipment in the adjacent control room.

The PC is connected to the nanoLIAB through the ethernet connection. The Valdemar dummy head receives the output via headphones, connected to the nanoLIAB. The outputs of Valdemar are connected, via a phantom power box, to the data acquisition platform, Harmonie.

The PC connects to the nanoLIAB user interface, with any type of web browser.

On the PC in the control room, the user interface will then be used to change wind speeds, source levels and distances. At the same time the Harmonie system will be used used to analyze the sound pressure levels recieved by the Valdemar dummy head.

Once the measurements are done, the measured output levels are compared to the levels they were supposed to be.

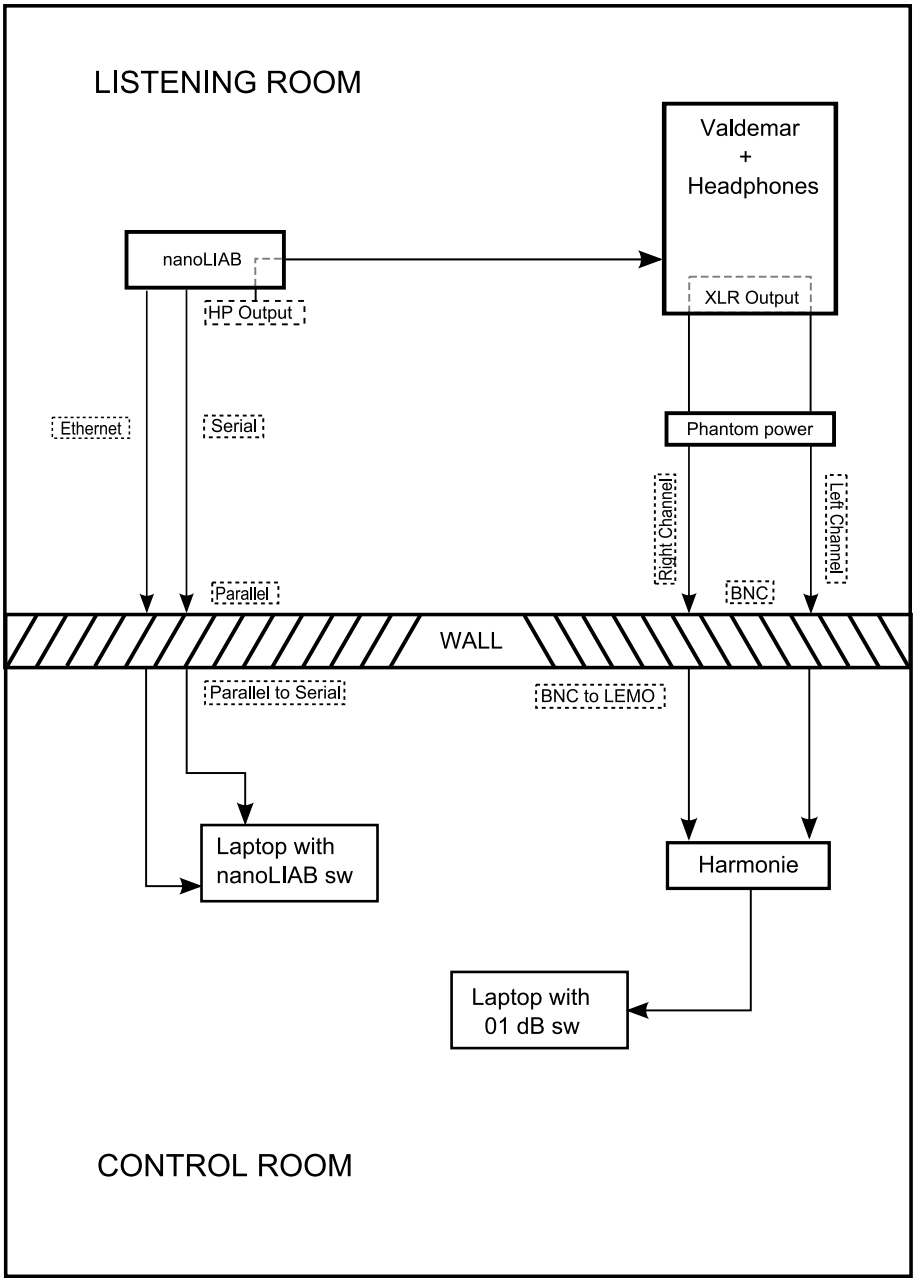

Figure 13.1: System test setup.

13.2 Test Results

As mentioned, there was no time to perform the described test, however a very simple test was conducted.

1. A USB disc with WAVE files and a PHP file with related data was connected to the nanoLIAB
2. A set of headphones was plugged in to the nanoLIAB
3. The nanoLIAB was turned on

4. A PC was used to connect to the web interface via a LAN cable

Once connected, various options were selected, while listening to the sounds in the headphones. By this simple test, it was clear that the sound levels changed almost instantly when the distance or wind speed was changed.

Also when changing the language setting, the language changed, so this part of the user interface also works.

14 Conclusions

In this thesis, a system that is able to reproduce the sound from noise sources and show the different levels present at different distances and wind speeds has been designed.

The current legislations and standards related to assessment of noise have been reviewed within the fields of wind turbines, roads, railways and airports.

To obtain the sound material, the noise from a wind turbine was measured and recorded, following the requirements of the current Danish legislation and relevant standards, such as the IEC 61400, as close as possible.

The aspects of the requirements that were not fulfilled were

- Measurement and recording of background noise – instead the percentile level, L_{90}, was used
- Minimum duration of measurement session at the required wind speeds

The data from the measurements has been analyzed according to relevant parts of the standards and the system requirements. As a result, relevant sound level measures, such as $L_{A,eq}$ and L_{DEN}, at various distances have been calculated

To implement the system a small computer called nanoLIAB was acquired. Around this device an entire system has been constructed by

- Changing the basic distribution in order to give adequate basic functionality
- Making a custom sound player for WAVE files
- Making a user interface that can be controlled via a normal web browser

The user interface provides a good overview of what is being played, by showing the user relevant data about the source that is being played and enables the user to change the settings and thereby playing a different sound.

Although the work has been focused on wind turbine noise, the system may apply to other sources, such as road noise, airport noise or railway noise.

The following aspects of the system was not completed in a satisfactory manner, due to time constraints

- The equalization of the system was not finished – instead a level calibration was done
- The system test was not executed. It has not been determined whether the system plays sound at the requested sound pressure level
- The user interface does not fullfill the full set of presentation requirements, however the basic functionality works

Despite the limited amount of measurement data and the lacks in the implementation of the system, the overall result is considered to be a success. The basic framework of the system has been implemented and all the limitations of the system are either missing implementation of almost finished designs or execution of measurements and experiments that have been defined.

From a learning perspective, the project has definitely been a succes, in the sense that many different aspects of acoustics engineering has been a part of the project. This includes understanding and following acoustic measurement standards and dealing with the inability to fulfill certain requirements by finding alternative solutions as well as doing specific audio engineering.

15
Future Perspectives

Future research in this topic may include:

- An improvement in the performance of the measurements. The acquisition of data from a wider range of wind speeds could not be achieved in this project. Having a large amount of data as recommended by the standard IEC 61400 would considerably enhance the appeal of the system It is specially advised to measure the background noise in the field. The background noise estimation by means of the percentile level L_{90} is accepted and suggested by standards dedicated to industrial noise in which the activity cannot be stopped. This could be the case for us as well, as the turbine could not by any means stopped. However, regarding the field measurements, this thesis sticks to the local legislation and the standard IEC61400 In the case that the turbine could not be stopped, it is preferable to choose a location in which a quiet area can be found around the turbine. This way, the measurements of the background noise could be performed at a point far enough from the turbine without the influence of it
- A way to deal with the background noise. It must be taken into account that the background noise plays an important role in the play back of the recorded sounds so it could be used either to "switch the turbine on and off" or to filter it out in order to replace it with a different background noise
- Filtering of the recorded signals in order to simulate the air absorption as a function of the distance. This could be done with better approximations than the ones given in the danish legislations
- Performing a proper equalization of the system instead of the simpler level calibration. This could include examining further combinations of mixer and WAVE file levels.
- A further investigation of the origin of the output level differences detected on the two channels of the system "nanoLIAB + headphones". The issue could be caused by several things. The output of the nanoLIAB, could be offset. The headphones could have different transfer functions. The microphones of the dummy head could be miscalibrated. And finally, the Harmonie system has an automatic gain setting for each channel.

- A proper test of the system should be done and documented more extensively. Due to time limitations this could not be done in this thesis although the system has been feature tested and a demonstration will be done during the presentation of this work
- Other noise sources such as road noise or airport noise should be tested on the system in order to assess the competence of the system in other areas apart from wind turbines. Of special interest are the two mentioned noise sources since they represent most of the annoyances caused on communities

Bibliography

[Age09] Danish Energy Agency. *Wind Turbines in Denmark*. Danish Energy Agency, 1 edition, 2009. ISBN 978-87-7844-821-7.

[ALR04] James F. Manwell Anthony L. Rogers. Wind Turbine Noise Issues. white paper, Renewable Energy Research Laboratory Center for Energy Efficiency and Renewable Energy. Department of Mechanical and Industrial Engineering. University of Massachusetts, 2004.

[Ass02] European Acoustics Association. Webportal of the european acoustics association. Internet, Feb 2002. `http://www.euracoustics.org/`.

[Ass03a] Danish Wind Industry Association. Guided Tour. Internet, September 2003. `http://www.windpower.org/en/knowledge/guided_tour.html`.

[Ass03b] Danish Wind Industry Association. Guided Tour on Wind Energy. Internet, july 2003.
`http://www.talentfactory.dk/en/tour/`.

[BK02] Brüel and Kjær. *B&K, Technical Documentation. 2238 Mediator Integrating Sound Level Meter. Frequency Analysis Software, BZ7123*, Feb 2002.

[BK04] Brüel and Kjær. Case Study: The Energy Research Centre of the Netherlands. Noise Measurements on Wind Turbines. *Brüel & Kjær*, 2004. `http://www.bksv.com`.

[Com00] International Electrotechnical Commision. IEC 60804: Integrating-averaging sound level meters. Standard, International Electrotechnical Commision, 2000.

[Com09] International Electrotechnical Commision. IEC 61400: Wind turbine generator systems. Standard, International Electrotechnical Commision, 2009.

[CVBG03] M.J. Carpena, J. Vera, S. Bleda, and E. Gimeno. Medidas de Aislamiento a Ruido Aereo en Cerramientos Arquitectonicos con Tecnicas MLS. *Tecniacustica Bilbao 2003*, 2003. internet; `http://www.sea-acustica.es/Bilbao03/aaq009.pdf`.

[DFTN99] David Formenti and Trevor Norsworthy. Microphone Selection. *Sound and Vibration*, 1999.

[dS] 01dB Stell. *dBBati32, Building Acoustics Analysis*. Internet; `http://www.sagetechnologies.com/principals/01db/pdf_documents/dbbati32.pdf`.

[fS79] International Organisation for Standardization. DS/ISO 3891: Procedure for describing aircraft noise heard in the ground. Standard, International Organisation for Standardization, 1979.

[fS03] International Organisation for Standardization. DS/ISO 1996: Description, measurement and assessment of environmental noise. Standard, International Organisation for Standardization, 2003.

[Gro] BSI Group. British standard, bs 4142: 1997 - rating industrial noise affecting mixed residential and industrial areas.

[IM91] IBM Corporation and Microsoft Corporation. Multimedia Programming Interface and Data Specifications 1.0. Specification, 1991. http://www-mmsp.ece.mcgill.ca/Documents/AudioFormats/WAVE/Docs/riffmci.pdf.

[Inc] Google Inc. Storvorde, Danmark. Google Maps. Internet. http://maps.google.com.

[KJPS06] Jørgen Kragh, Hans Jonasson, Birger Plovsing, and Ari Sarinen. User's guide nord2000 road. Technical report, Delta, May 2006.

[Kom94] Aalborg Kommune. Lokalplan 15-022: Vindmøller ved sømærkevej rærup. Technical report, December 1994.

[KP09] Stylianos Kephalopoulos and Marco Paviotti. Advancement in the development of european common noise assessment methods: where are we? *Euronoise Conference. Edinburgh, Scotland*, October 26-28, 2009.

[LIA06] LIAB ApS. *nanoLIAB User's Manual*, November 2006.

[LIA07] LIAB ApS. *A Sound System for the nanoLIAB*, May 2007.

[Mil94a] Miljøministeriet. Støj fra flyvepladser. Vejledning fra Miljøministeriet, November 1994.

[Mil94b] Miljøministeriet. Vej nr 60274 af 01/11/1994: Støj fra flyvepladser. Internet, November 1994. Danish laws on noise action plans. https://www.retsinformation.dk/Forms/R0710.aspx?id=12148.

[Mil97] Miljøministeriet. Vejledning fra miljøstyrelsen nr. 1 1997: Støj og vibrationer fra jernbaner. Internet, 1997. Danish laws on noise action plans. http://www.mst.dk/NR/rdonlyres/BA55FAB0-D015-48D6-B3D3-23CD081D4F5D/0/tillaeg_til_togstoejvejledning_end.pdf.

[Mil06a] Miljøministeriet. Bek nr 1518 af 14/12/2006: Bekendtgørelse om støj fra vindmøller. Internet, Dec 2006. Danish laws on wind turbines. https://www.retsinformation.dk/Forms/R0710.aspx?id=13020.

[Mil06b] Miljøministeriet. Vejledning fra miljøstyrelsen nr. 4 2006: Støjkortlaegning og støjhandlingsplaner. Vejledning fra Miljøministeriet, November 2006. Danish laws on noise action plans. http://www2.mst.dk/Udgiv/publikationer/2006/87-7052-146-8/pdf/87-7052-146-8.pdf.

[Mil07a] Miljøministeriet. Støj-danmarkskortet. Internet, 2007. http://www.mst.dk/Virksomhed_og_myndighed/Stoej/kortlaegning_af_stoej/stoej_dk_kort/.

[Mil07b] Miljøministeriet. Vejledning fra miljøstyrelsen nr. 4 2007: Støj fra veje. Internet, 2007. Danish laws on noise action plans. http://www.mst.dk/NR/rdonlyres/951BDC20-4089-4425-9506-3B760DB860A2/48496/Vejstjvejledning2.pdf.

[Org95] World Health Organization. Guidelines for community noise. *Stockholm University and the Karolinska Institute*, 1995. http://www.who.int/docstore/peh/noise/guidelines2.html.

[Org05] International Standard Organisation. *EN ISO 3095:2005, Railway applications - Acoustics - Measurement of noise emitted by railbound wehicles*. International Standard Organisation, 1 edition, 2005.

[otEC01] Official Journal of the European Communities. Commission Directive EC/2001/16 of 19 march 2001 on the interoperability of the trans-european conventional rail system. Report, The European Parliament and the Council of the European Union, 2001.

[otEC02a] Official Journal of the European Communities. Commission Directive EC/2002/30 of the european parliament and of the council of 26 march 2002 on the establishment of rules and procedures with regard to the introduction of noise-related operating restrictions at community airports. Report, The European Parliament and the Council of the European Union, 2002.

[otEC02b] Official Journal of the European Communities. Commission Directive EC/2002/49 of 25 june 2002 relating to the assessment and management of environmental noise. Technical report, The European Parliament and the Council of the European Union, 2002. internet; http://europa.eu/legislation_summaries/environment/noise_pollution/l21180_en.htm.

[otEC07] Official Journal of the European Communities. Commission Directive EC/2007/34 of 14 june 2007 amending, for the purposes of its adaptation to technical progress, council directive 70/157/eec concerning the permissible sound level and the exhaust system of motor vehicles. Report, The European Parliament and the Council of the European Union, 2007.

[OZ07] Monte Ohrt and Andrei Zmievski. Smarty manual. Internet, September 2007. http://www.smarty.net/distributions/manual/en/Smarty-2.6.14-docs.pdf.

[Ped07] Eja Pedersen. Human response to wind turbine noise, 2007. ISBN 978-91-628-7149-9. http://cvi.se/uploads/pdf/Kunskapsdatabas%20miljo/Ljud%20och%20Skuggor/Ljud/Forskningsresultat/Pedersen_Thesis.pdf.

[Pot08] Doug Potter. How to daemonize in linux, 2008. http://www-theorie.physik.unizh.ch/~dpotter/howto/daemonize.

[Rab04] André Rabold. Smarty multilanguage support. Internet, January 2004. http://smarty.incutio.com/?page=SmartyMultilanguageSupport.

[sgm08] N-ALM Noise sub-group meeting. Survey on background information of aircraft noise calculation methodologies and practices in nordic countries. *28 Jan 2008, assembled by Finavia*, 2008.

[TRs] Tontechnik-Rechner-sengpielaudio. Weighting filter after din en 61672-1 2003-10 (din-iec) dba and dbc - the difference. http://www.sengpielaudio.com/calculator-dba-spl.htm.

[TS00] Jeff Tranter and Hannu Savolainen. Open sound system – programmer's guide, 2000. http://www.opensound.com/pguide/oss.pdf.

[Vej02] Vejdirektoratet Vejregeludvalget. 2.30.02 omgivelserne. støj. støjhensyn ved nye vejanlæg. Technical report, February 2002.

[Weaa] Heavy Weather. *Bedienungsanleitung, Proffesionelle Funk-Wetterstation*. Internet; http://www.heavyweather.info/new_german/2300pdf/deutsch_Hardware_Manual.pdf.

[Weab] Heavy Weather. *Kurzanleitung für die Inbetriebnahme der Wetterstation*. Internet; http://www.heavyweather.info/new_german/2300pdf/deutsch_Quick_Set_Up.pdf.

List of Figures

List of Tables

Part VI

Appendix

A

Wind Turbine Measurement Procedure

This appendix gives the detailed information about the requirements and formulas used for wind turbine noise measurements given by Announcement no. 1518 from the Danish Ministry of the Environment [Mil06a]. The details of the measurement requirements are found in Appendix 1 of the Announcement and states that measurements carried out according to IEC 61400 fulfill the requirements and may be used for calculating the apparent sound power level, $L_{\text{WA,ref}}$.

Once the sound power level is known, it can be used for calculating the sound pressure level at a given position, L_{pA}, which is the level that is used for noise limits in the public regulations.

The basic measurement requirement regarding distances and angles between turbine and equipment are illustrated on Figure A.1.

The procedure for recording and calculating the noise is as follows.

1. Record the noise of the wind turbine
   - At a standardized distance, $R_0 = h + \frac{d}{2} \pm 20\%$
   - With a microphone on a reflecting plate of minimum dimensions $1 \cdot 1$ m on the ground with a wind hood to avoid noise from the wind
   - The direction from tower to microphone must not deviate more than $\pm 15^\circ$ from the direction of the wind
   - Noise is measured as a number of A-weighted octave spectra from 63 to 8000 Hz – each averaged over at least 1 minute
   - At least 5 spectra are measured with 5.5 m/s $\leq v_{\text{ref}} \leq 6.5$ m/s (at least one in the range below 6.0 m/s and one above)
   - At least 5 spectra are measured with 7.5 m/s $\leq v_{\text{ref}} \leq 8.5$ m/s (at least one in the range below 8.0 m/s and one above)
   - The A-weighted reference spectra are calculated as energy means of the measured spectra
2. Normalize to a wind speed in the reference height of 10 m

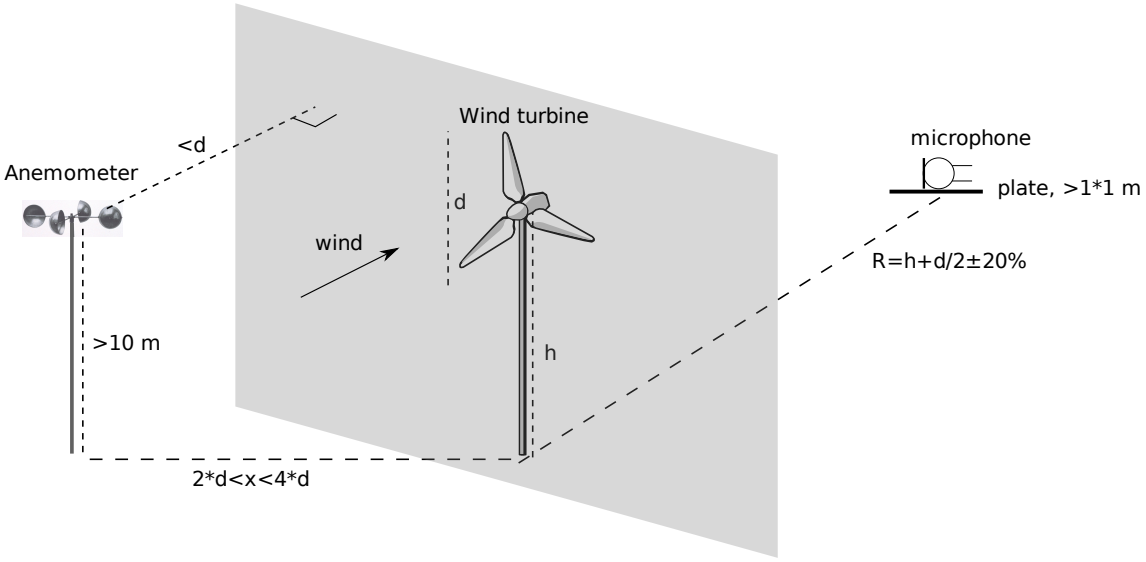

Figure A.1: Sketch of the measurement requirements of the wind turbine announcement[Mil06a]

- If effect curves of the wind turbine are known, use these to calculate the wind speed at rotor height and compensate for this height to find v_{ref}, using equation A.1
- Otherwise, measure the wind speed at a representable position, in the opposite direction of the wind
  - At a distance, x, from the turbine $2 * d \leq x \leq 4 * d$
  - With the height of the anemometer at least 10 m
  - Not diverting more than $\pm d$ from the vertical plane of the rotor - see Figure A.1
  - According to the terrain, different roughness from table A.1 are used
  - Calculation of v_{ref} is done using equation A.2

3. Correct for background noise to achieve $L_{\text{A,ref,k}}$
   - Similar noise measurements are performed with the wind turbine stopped (same requirements to spectra and v_{ref})
   - The wind is measured in the same way as above
   - The total level of background noise, $L_{\text{A,eq}}$, must be at least 6 dB lower than the total level of the turbine noise, otherwise a new measurement is required, when the background noise is lower.
   - In control of the noise impact, the 6 dB difference requirement can be loosened if the calculated level after a correction of background noise of -1.3 dB is no higher than the specified limits[1]
   - The correction for background noise is calculated using equation A.3
4. Determine the sound power level of the turbine $L_{\text{WA,ref}}$
   - $L_{\text{WA,ref}}$ is calculated using equation A.4
5. Determine the sound pressure level at a given position, L_{pA}

[1] the specified limits refer to the limits set by law or other requirements – in this case the wind turbine announcement[Mil06a]

(a) Air absorption, ΔL_α, is calculated using equation A.5

(b) Sound pressure level in 1/1 octave bands, L_{pA}, is calculated using equation A.6

(c) The total sound pressure level, $L_{\text{pA,tot}}$, is calculated using equation A.7

6. Determine audible tonality and noise impact, L_r

   - If a frequency analysis shows a tonality of the turbine noise, a penalty of 5 dB is added to the calculated sound pressure level

7. In the case of multiple turbines, the following applies

   - The noise measurements are performed on at least 3 random turbines ($\pm$2-3 dB difference of L_{WA} can be expected for identical types)
   - For the remaining turbines, the energy mean of the three measured turbines
   - The total sound pressure level of the turbine group is calculated according to equation A.10

In the following all the equations for calculating the above are described.

Correction of wind speeds for turbines with known power curve

The power curve is a graphical presentation of the relation between wind speed at the turbine hub and the power output from the turbine. When knowing the output power and the height, the wind speed can therefore be found and the speed in the reference height (10 m) can then be calculated.

$$v_{\text{ref}} = v_h \cdot \frac{\ln\left(\frac{z_{\text{ref}}}{z_{0\text{ref}}}\right)}{\ln\left(\frac{h}{z_{0\text{ref}}}\right)} \tag{A.1}$$

where

v_{ref}	is the reference wind speed	[m/s]
v_h	is the wind speed at the turbine (read from power curve)	[m/s]
h	is the height of the turbine hub	[m]
z_{ref}	is the reference height of 10 m	[m]
$z_{0\text{ref}}$	is the reference roughness of 0.05 m	[m]

Correction of wind speeds for turbines when power curve is not known

When the above mentioned power curve or the power output is not known, the wind speed must be measured as desribed ealier and the wind speed at reference height 10 m are then calculated.

$$v_{\text{ref}} = v_z \cdot \frac{\ln\left(\frac{z_{\text{ref}}}{z_{0\text{ref}}}\right) \cdot \ln\left(\frac{h}{z_0}\right)}{\ln\left(\frac{h}{z_{0\text{ref}}}\right) \cdot \ln\left(\frac{z}{z_0}\right)} \tag{A.2}$$

where

v_z	is the wind speed at the anemometer	[m/s]
z	is the height of anemometer	[m]
z_0	is the roughness of the terrain, found from table A.1	[m]

Type of terrain	Roughness, z_0 [m]
Water, snow sand	0.0001
Open flat landscape, bare soil, mown lawns	0.01
farm land with vegetation	0.05
residential area, smaller towns, areas with dense, high vegetation	0.01

Table A.1: Roughness of various types of terrain

Correction for background noise

The background noise is measured under similar conditions as the wind turbine, with the turbine stopped, and the corrected level is then calculated.

$$L_{\text{A,ref,k}} = 10 \cdot \log\left(10^{\frac{L_{\text{A,ref}}}{10}} - 10^{\frac{L_{\text{A,b}}}{10}}\right) \tag{A.3}$$

where

$L_{\text{A,ref,k}}$	is the corrected reference sound pressure level in 1/1 octave bands	[dB]
$L_{\text{A,ref}}$	is the meaned noise sound pressure level in 1/1 octave bands	[dB]
$L_{\text{A,b}}$	is the meaned background noise sound pressure level in 1/1 octave bands	[dB]

Sound power level of the turbine

When the background noise correction is done, the sound level is calculated back to the source to estimate the sound power level output from the turbine, using the distance law and correcting for the baffle on the ground.

$$L_{\text{WA,ref}} = L_{\text{A,ref,k}} + 10 \cdot \log\left(4\pi \cdot \left(R^2 + h^2\right)\right) - 6\text{dB} \tag{A.4}$$

where

$L_{\text{WA,ref}}$	is output sound power of the turbine in 1/1 octave bands	[dB]
R	is the distance from the microphone to the turbine base	[m]
6dB	is a correction for the reflecting plate on the ground	[dB]

Sound pressure level at a given distance

Once the sound power level of the turbine is determined, the level can be calculated at any given distance by taking air absorption and the distance law into account.

$$\Delta L_\alpha = \alpha \cdot \sqrt{l^2 + h^2} \tag{A.5}$$

$$L_{\text{pA}} = L_{\text{WA,ref}} - 10 \cdot \log\left(l^2 + h^2\right) - 11\text{dB} + \Delta L_g - \Delta L_\alpha \tag{A.6}$$

$$L_{\text{pA,tot}} = 10 \cdot \log\left(\sum 10^{\frac{L_{\text{pA}}}{10}}\right) \tag{A.7}$$

where

ΔL_α	is the air absorption in 1/1 octave bands	[dB]
α	is the air absorption coefficients seen in table A.2	[dB/m]
l	is the distance from the turbine base to the calculation point	[m]
h	is the height of the turbine hub	[m]
L_{pA}	is the calculated sound pressure level at the position in 1/1 octave bands	[dB]
ΔL_g	is a correction for the terrain: 1.5 dB for land based turbines and 3 dB for offshore turbines	[dB]
11dB	is a correction for distance ($10 \cdot \log(4\pi)$)	[m/s]
L_{pA}	is the total calculated sound pressure level at the position	[dB]

The uncertainty of this method is ± 2 dB

Octave band center frequency in Hz	63	125	250	500	1000	2000	4000	8000
α_a in dB/m	0.0001	0.0004	0.001	0.002	0.0036	0.0088	0.029	0.1045

Table A.2: Air absorption coefficients at a relative air humidity of 80% and an air temperature of 10°C

Noise impact

If the noise contains audible single tones, a 5 dB penalty is added, because audible tonality is more annoying than a flat spectrum of sound:

$$L_r = L_{\text{pA,tot}} + 5 \tag{A.8}$$

otherwise

$$L_r = L_{\text{pA,tot}} \tag{A.9}$$

Sound pressure level at a given distance for multiple turbines

When considering multiple turbines, the above equations are applied to calculate the total pressure contribution for each turbine at the wanted position, to find the combined total level.

$$L_{\text{total}} = 10 \cdot \log\left(10^{\frac{L_{p1}}{10}} + 10^{\frac{L_{p2}}{10}} + \cdots\right) \tag{A.10}$$

where

L_{total}	is the total calculated sound pressure level at the position	[dB]
L_{p^i}	is the calculated sound pressure contribution at the position from the i'th turbine	[dB]

B

Measurement Journal

In this appendix a full journal of the measurements is included. All measurements were carried out in order to acquire all practical data necessary for the realization of this Msc. Thesis. The structure of the journal is as specified by the standar IEC 61400-11 *Wind Turbine Generator Systems, Part 11: Acoustic Noise Measurement Techniques*. Data acquisition and equipment used also fulfill all the requirements given by this standard.

B.1 Characterization of the wind turbine

Table B.1 includes the following details from the wind turbine. The turbine was manufatured on 1988 and, due to its age, it is hard to find all data required by the standard IEC 61400-11. Thus, all data included sticks to all that can be found.

Wind turbine details	
Manufacturer	Danish Wind Power (DWP)
Model Number	D150S
Serial Number	570715000000007859
Operating details	
Turbine axis position	Horizontal
Rotor orientation	Upwind
Hub height [m]	30.7
Distance rotor center-tower axis	
Diameter of the rotor [m]	22.2
Tower type	Tube
Power control	Stall
Rated Power Output [kW]	150
Rotor details	
Blade type	LM
Number of blades	3

Table B.1: Details of the Danish Wind Power D150S wind turbine measured for the realization of this thesis

B.2 Physical environment

Measurements were carried out in Engvej Rd., Storvorde, a small town to the east of Aalborg. *Figure B.1* shows the site map including a the exact location.

Figure B.1: Location of the wind turbines [Inc]

The measurement area was located on a farmland. The wind turbines stood on the crops. The complex consisted of a group of three horizontally aligned mills. The area showed a flat terrain crossed by a non-paved road featured by the presence of gavel. A stream of stagnant water next to the road and some vegetation such as bushes and small trees characterized the landscape. The impossibility of stopping two of them and the exceptional location of them (far enough from any other noise sources such as roads, highways, railways . . .) obliged to measure the three mills at once. To minimize the effects of two wind mills and obtain most of the performance of a single wind mill, measurements were carried out on the wind mill placed on one of the edges of the line formed by them. No further reflecting surfaces were present on the area. *Figure B.2* shows the alignment of the mills.

Figure B.2: Alignment of the wind turbines [Inc]. The dotted line shows the measurement area

Figures B.3, B.4 show the area respectively from the wind measurement mast position and the measurement measurement position.

Figure B.3: Wind turbine view from the anemometer position

Figure B.4: Wind turbine view from measurement microphone position

Figure B.5 shows the measurement board positioned on the ground as specified on the standard IEC 61400-11.

Figure B.5: Measurement microphone position

B.3 Instrumentation

A considerable amount of equipment is necessary to undertake a wind turbine noise assessment. *Figure B.2* lists all equipment, software and supplementary equipment needed for an ordinary assessment of wind turbine noise. *Figure B.3* includes, though, the necessary equipment for the measurement of weather conditions, with special emphasis on the measurement of wind speed and direction. The anemometer position follows the rules given by the standard IEC 61400.

Item	AAU No.
B&K 4165 1/2", free field microphone	07954
B&K 4165 1/2", free field microphone	07955
B&K 2238 Mediator, sound level meter	33948
B&K 4230, calibrator	08373
B&K 4231, calibrator	78301
Grass 26AK, preamplifier	52664
Grass 26AK, preamplifier	52665
Roland-Edirol R-09, Digital recorder	64676
01 dB Harmonie, data acquisition system	04124
Laptop HP Omnibook 6000	47220
Software and supplementary equipment	
dB Trig32, data acquisition software 01 dB	
Base plate	
LEMO, connection cables	
1/2 wind screen	
Calibration requirements	
All equipment was calibrated prior to any measurement	

Table B.2: Equipment needed to assess wind turbine noise

The measurement height is 10 meters. The distance from the rotor center of the wind turbine at which the anemometer shall be mounted must be in between 2 and 4 times the diameter of the circunference formed by the blades. In this case, minimum and maximum distances are 44.4 and 88.8 meters. The distance for the placement of the anemometer was 46 meters. Wind conditions have to be measured in the upwind direction within the ragen given by an angle, β, calculated from

Item	AAU No.
Weather station with anemometer	2157-45
Wind measurement mast	
Laptop HP Omnibook 6000	47220
Software and supplementary equipment	
Heavy Weather 2.0, acquisition software	
Wind measurement details	
Anomoterer Position	see *Section B.4*
Measurement height [m]	10

Table B.3: Equipment and planning needed to assess weather conditions

Equation B.1.

$$\beta = \frac{z - z_{\text{ref}}}{H - z_{\text{ref}}}(\beta_{\text{max}} - \beta_{\text{min}}) + \beta_{\text{min}} \tag{B.1}$$

where

z	is the anemometer height	10 [m]
z_{ref}	is the reference height	10 [m]
H	is the height of the rotor centre of the wind turbine	30.7 [m]
β_{max}	is the maximum angle for β	90°
β_{min}	is the minimum angle for β	30°

Figure B.6 illustrates the region where the mast can be placed for wind measurement purposes. Wind results obtained after the measurements are presented on *Section B.5*.

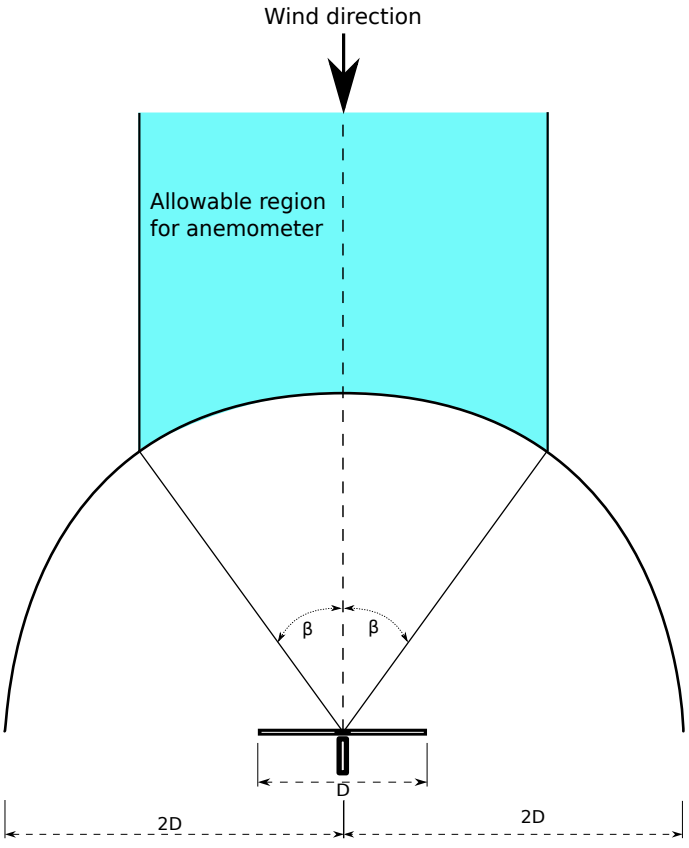

Figure B.6: Allowable region for mast allocation

B.4 Acoustic data

The standard IEC 61400 specifies a measuring position according to *Equation 1*. This implies the possibility of measuring at different range of positions. This range varies within a $\pm 20\%$ from a reference which in this case is 51.8 meters (see *Figure B.7*). Due to limitations of different nature (see *Section 5.1*), the realization of the measurements did not allow to measure in more than one position.

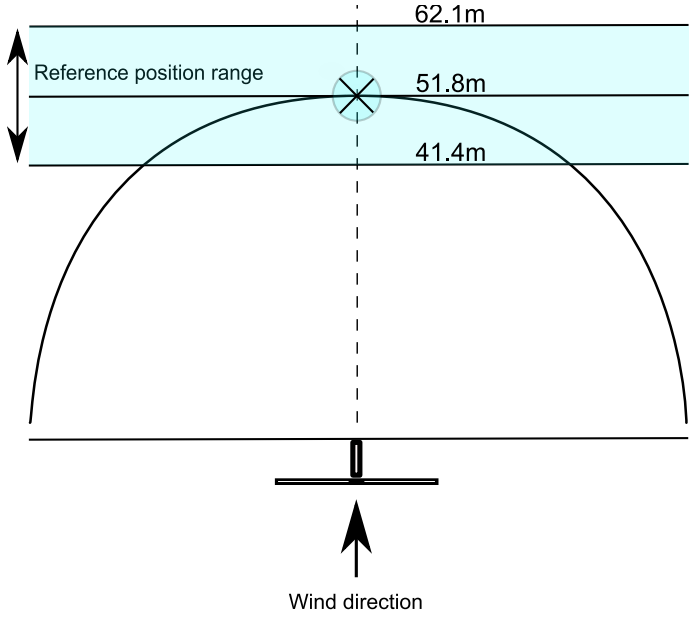

Figure B.7: Reference position of the measurement microphone

Apparent Sound Power Levels Sound power levels at each integer wind speed from 6 to 10 m/s shall be reported. Unfortunately, such speeds could not be obtained during the measurement session. *Figures B.8, B.9* and *B.10* show the different results obtained for the different wind speeds present at the time of the measurements.

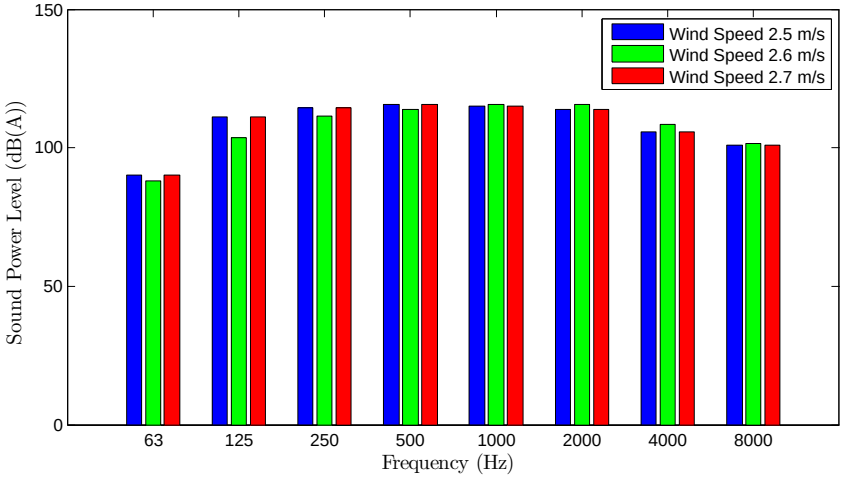

Figure B.8: Apparent Sound Power levels for wind speeds 2.5, 2.6 and 2.7 m/s

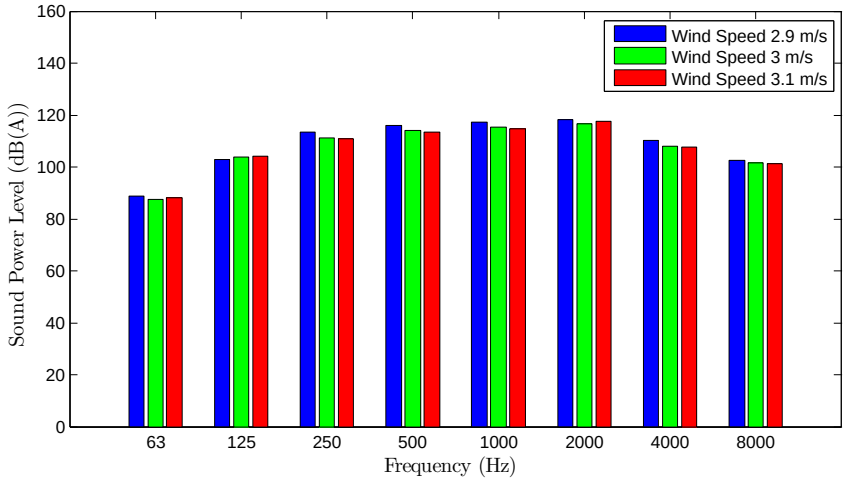

Figure B.9: Apparent Sound Power levels for wind speeds 2.9, 3 and 3.1 m/s

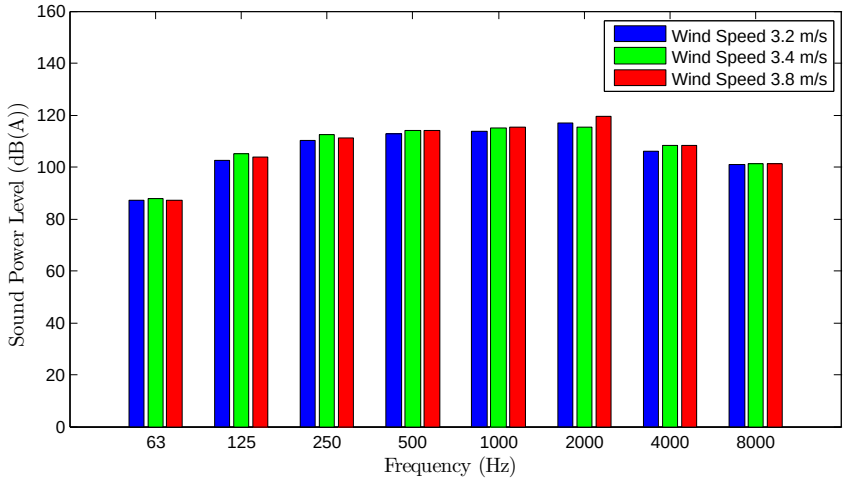

Figure B.10: Apparent Sound Power levels for wind speeds 3.2, 3.4 and 3.8 m/s

The graphs show very similar estimations of the apparent sound power level of the turbine due to the very small variation of the wind.

Measured data Another aspect of interest in the measurement journal is the comparison of the background noise measured and the wind turbine noise measured together with the background noise. *Figure B.11* shows this comparison. Unfortunately, the background noise could not be measured during the session so it had to be estimated. This was done by means of the percentile level L_{90}, an indicator to express the level that is exceed 90% of the time within a measure. This fact makes this comparison not as interesting as it could be when measuring the background noise.

Sound pressure spectrum The following Figures plot the sound pressure spectra of the measured wind turbine noise after the background noise correction. Data is presented in one-third octave bands for all measured wind speeds.

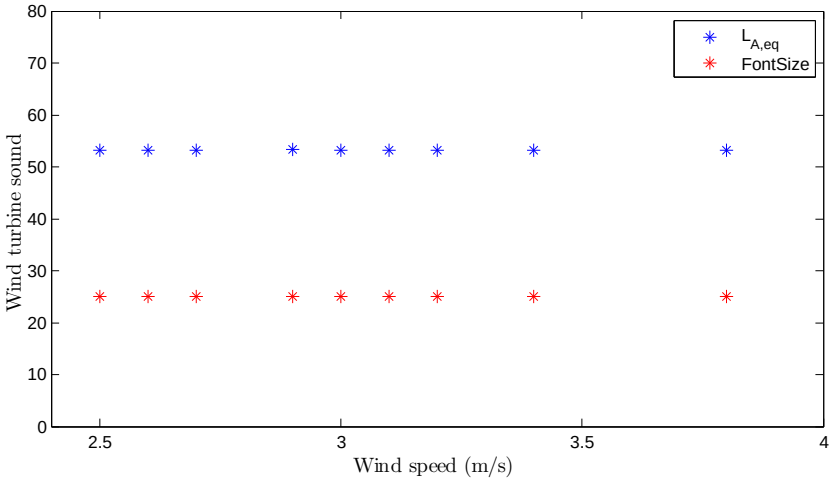

Figure B.11: Measured data pairs at reference position

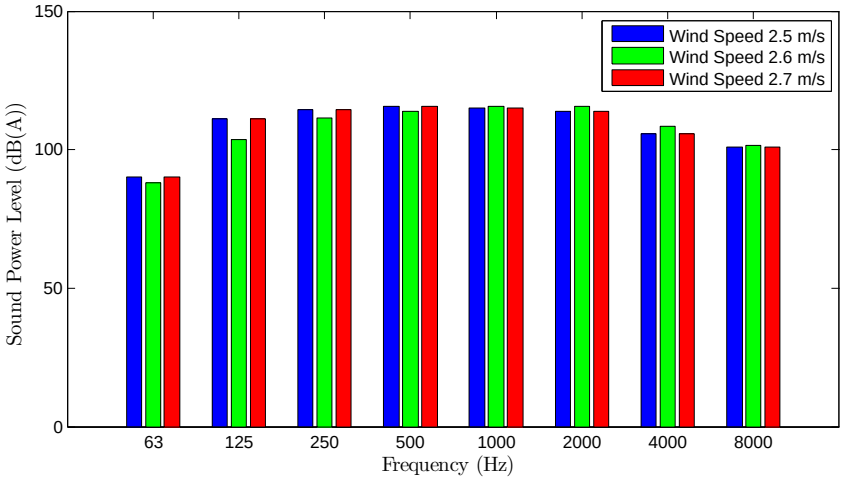

Figure B.12: Sound pressure levels for wind speeds 2.5, 2.6 and 2.7 m/s

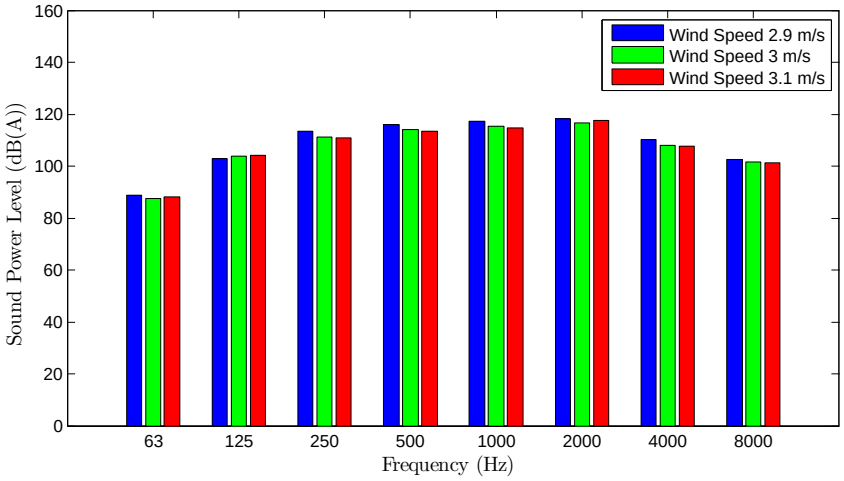

Figure B.13: Sound pressure levels for wind speeds 2.9, 3 and 3.1 m/s

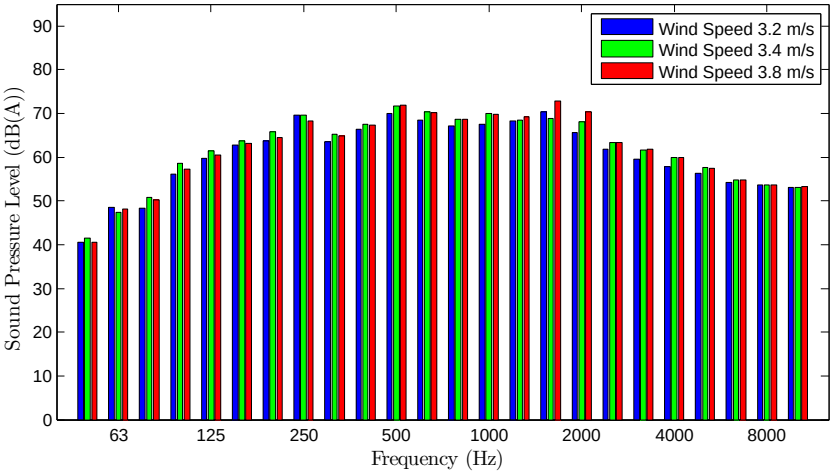

Figure B.14: Sound pressure spectra for wind speeds 3.2, 3.4 and 3.8 m/s

B.5 Non-acoustic data

The wind speed determination method is the method referred in the standard IEC 61400 as *Method 2: determination of wind speed with an anemometer*. Wind measurement results were adjusted to a height of 10 meters and a roughness length of $z_0 = 0.05m$ as the terrain corresponded to a farmland with some vegetation. The estimation of the roughness length is obtained according to *Table A.1* which specifies the different values available regarding the type of terrain. This table is also given in the standard IEC 61400. *Table B.4* shows the all relevant weather data collected during the measurement session. Wind speed and direction, air temperature, atmospheric pressure.

Minute no.	Wind Speed (m/s)	Wind direction	Air temperature (◦C)	Atmospheric pressure (%)
1	2.7	E	12	38
2	2.9	ESE	12	38
3	2.9	ESE	12	38
4	2.6	ESE	12	38
5	2.6	ESE	12	38
6	3.8	E	12	38
7	3.0	E	12	38
8	3.0	E	12	38
9	3.2	ESE	12	38
10	3.2	ESE	12	38
11	3.2	ESE	12	38
12	3.1	ESE	12	38
13	3.1	ESE	12	38
14	3.1	ESE	12	38
15	2.7	SE	12	38
16	2.7	ESE	12	38
17	3.4	ESE	12	38
18	3.4	ESE	12	38
19	2.5	ESE	12	38

Table B.4: Non acoustic data collected on wind turbine noise measurement session

C

Development Tools

In this appendix, the tools and scripts used for the development of the system will be described.

The main part of development has taken place on computers with Ubuntu Linux operating system. Since crosscompiler is made for Linux and the software has been written to work on Linux, using Linux during development, also allowed testing code without transferring it to the nanoLIAB. The development systems utilize a Bash shell, and a number of shell scripts have been written for this to aid in the development and easy reproduction of generating the various parts of the project.

C.1 Installing the Cross Compiler

As mentioned in *Chapter 10*, the supplied cross compiler had some issues, so it was necessary to recompile it from scratch.

The cross compiler used for building the nanoLIAB ARM binaries is a patched version of GCC compiled by using a script with accompanied patches called crosstool, which is available at `http://kegel.com/crosstool/`. The crosstool script is used to download, patch, build, and test various versions of gcc with different glibc versions for a number of different architectures.

The architecture of the nanoliab is “arm-softfloat” (determined by looking at the shipped cross compiler), and it was chosen to use GCC-3.4.5 with glibc-2.3.6, which are slightly newer than the versions offered by LIAB, however different versions would probably work.

A script was written to deal with a number of issues in relation to the compilation. It was necessary to install a number of extra tools which were required. As well as add a few extra patches/hacks to the configure scripts and a Makefile of glibc. The configure script for glibc had to be to modified to properly understand the versions of binutils and GCC installed on the computer and a Makefile had a quotation error, which meant that a generated file, *version-info.h*, had syntax errors.

The script, *install-crosscompiler.sh*, is available on the attached CD in the *Software* folder.

C.2 Make Base Script

In order to achieve all the tasks involved in creating the base distribution, a script was written.

The script, *make_base.sh*, is available on the attached CD in the *Software* folder and can be used to perform the following tasks:

- Download and patch PHP and THTTPD
- Compile PHP and THTTPD
- Temporarily install the THTTPD binary on the nanoLIAB for testing (requires the system to be connected)
- Set up and create a nanoLIAB disc image
- Clean up

In the beginning of the script, a number of settings are defined, including the IP address, and wanted password of the nanoLIAB.

When executing the script, the following arguments are understood:

- php – download, patch and compile PHP
- thttpd – download, patch and compile THTTPD with PHP support (requires PHP to be compiled)
- liab – setup, modify and create disc image (requires PHP and THTTPD to be compiled)
- all – equivalent to "php thttpd liab"
- install – temporarily install the compiled THTTPD binary on the nanoLIAB (requires PHP and THTTPD to be compiled)
- clean – remove all installed, compiled and extracted files
- --nobuild – skip compiling PHP and/or THTTPD
- --nodownload – skip downloading PHP and/or THTTPD
- --nopatch – do not apply patches to PHP and/or THTTPD

When making a disc image, the size of the image will be calculated, and compared to the maximum size available, thus ensuring that it is not too large. When an image has been created, a guide describing how to upload the image to the nanoLIAB is showed.

C.3 Make Web Package Script

For easily making a "distributable packet" that can be put on the system, a script called *make_web_package.sh* was made.

The script, which is available in the *Software* folder on the attached CD, copies the relevant files of the web interface and template system to a "web package" folder, and generates the *noise_setup.sh* script, which is automatically run when the system boots.

The script can be run with the following arguments:

- -i – install the interface on the LIAB using telnet and RCP
- -p package directory – use a different directory for putting the package in
- -v – verbose: get detailed execution information

C.4 Install Web UI Locally

In order to make it easier to program the web interface, a script was made that can install the interface on the local PC.

The script *install-web-local.sh*, which is also available in the *Software* folder on the CD, sets up the local Linux PC with apache to run the web interface on `localhost/liab` and allows the webserver to play audio, since this is not normally allowed. The interface can also be accessed by other users, by using the computer's IP.

The script can be run with the following arguments:

- -i – install the interface and setup audio access (default with no arguments)
- -u – uninstall the web interface and remove the audio access again
- -d www-directory – use a different directory for installing the interface
- -v – verbose: get detailed information

D

Noiseplay Program

This appendix contains a detailed overview of the functions of the Noiseplay program, including a description of how they work and a definition of input and output of each function.

In *Chapter 11*, the main design of the program is defined, and the functionality was divided into the following groups, which will be expanded into the functions below.

- Noiseplay – The main function and argument handling, which also includes every other part

  files: noiseplay.c, noiseplay.h

- Global settings and definitions

  files: global.h

- Daemon – functions for turning a process into a daemon

  files: daemon.c, daemon.h

- Sound – Soundcard interface functions: setup, buffering and playback

  files: sound.c, sound.h

- WAVE – functions for reading, checking and processing WAVE files

  files: wav_functions.c, wav_functions.h

- Settings – Functions for setting and changing the playback settings

  files: settings.c, settings.h

On *Figure D.1* a flowchart of the interaction between the all the functions of the program can be seen in relation to which . As it can be seen on the figure, most of the functions are called from the main program and only the settings functions and WAVE functions.

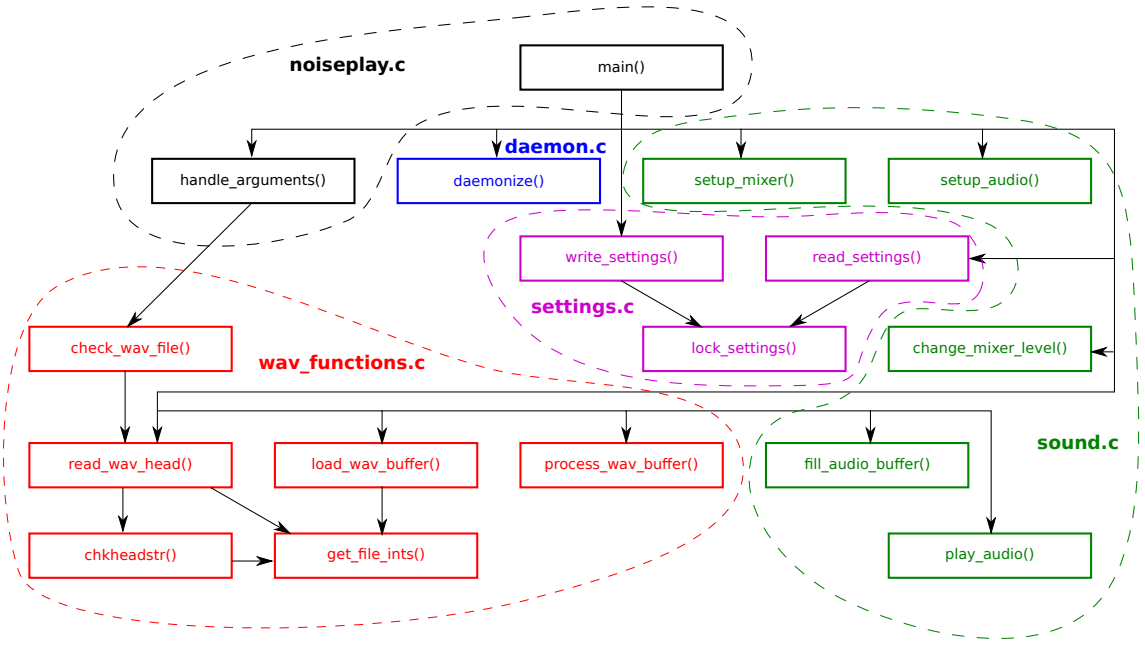

Figure D.1: Flowchart of the various functions and how they are called.

D.1 Noiseplay

Noiseplay contains the main function of the program, which includes all the other parts and therefore has a structure as outlined on *Figure 11.1* on page 87. The function used for handling the input arguments is also a part of this group.

Files: noiseplay.c and noiseplay.h

Functions: `main()`, `handle_arguments()`

D.1.1 main()

The main function of the noiseplay program. This function contains the controlling functionality and calls almost every major function of the entire program.

Input: Variable arguments. A filename is required as the first argument.

filename – name and path of the WAVE file to play (required)

[-g | --gain] gain – Change the gain of the file. the gain should be a floating point number

[-m | --mixer] level – Change the level of the hardware mixer. The level must be between 45 and 94 to stay in the linear range of the mixer gain.

[-p | --plays | -l | --loops] number of plays – Change how many times the file playback should be repeated. Default is infinitely (-1).

[-d | --debug | --no-daemon] – Debug flag. If set, debugging messages will be output.
A new process will remain in the foreground (not go in daemon mode).
A running (daemon) process will start to log messages to */var/log/noiseplay.log*.

Output: Success, Warning or Error. An error occurs if no filename is given or the first argument is not a valid WAVE file. A warning is showed if an argument is not recognized, however the

program will continue, ignoring the errornus argument.

The program path of the main function is to first handle the input arguments and then go into daemon mode before writing the command line settings to a file. The next step is to setup the mixer and audio devices properly, and finally the WAVE file from the input argument is played in a loop. Whenever a new instance of the program is started, the settings are saved in a file before the new instance exits, and this file will then be read from the loop, changing the settings if possible.

D.1.2 handle_arguments()

Parse the command line arguments into a settings struct as defined in settings.h.

Input: the variable arguments from main.

Output: a settings struct with the parsed arguments.

Errors: exits with error messages if

- No filename is given
- The first argument is not a valid WAVE file

Warnings: Shows a warning if an argument is not recognized

This function first validates that the first argument is a filename, and that the file is a WAVE file of a proper format. When this is done, the remaining arguments are looped over and parsed one by one. As the validation and parsing progresses, the data is saved in a settings struct.

D.2 Global settings and definitions

The global settings and definitions, that do not concern only one function or are relevant to have in a single place are saved in a single header file.

files: global.h

The fixed settings are amongst others the paths for the lock and log files and the size of the audio buffers.

D.3 Daemon

The daemon contains a function for turning a process into a daemon. The function returns the status of the daemonization to the child, while the adult is terminated inside the function.

Files: daemon.c and daemon.h

Functions: `daemonize()`

D.3.1 daemonize()

Function that turns the program into a daemon

Input: int debug – a flag which tell whether or not to go into daemon mode and redirect stdin, stdout and stderr.
TRUE, the program isn't actually turned into a daemon and messages are printed to stdout/stderr.
FALSE, turn into a daemon and re direct stdout/stderr to log files.

Output: Success, Failure or Process_locked.
SUCCESS if this is the only running instance, and the program has successfully been put in the background.
FAILURE if something went wrong after the parent exited.
PROCESS_LOCKED if another instance is already running

When the daemonize function is called, the execution is quite dependent on whether the debug flag is set. If the debug flag is **not** set, then the program will first go into the background, using `fork()` and then attempt to lock the process file. If the file is locked, the process will return with the output PROCESS_LOCKED. If the file is not locked, then the final task is to detatch properly from the starting tty and redirect stdin, stdout and stderr to /dev/null or log files, after which SUCCESS is returned.

In the case where the debug flag is set, apart from giving output messages, the program will avoid going into the background, and only the process locking will take place.

D.4 Sound

Soundcard interface functions used to setup the audio devices, convert a 16 bit buffer to the sound card format and playback the buffer. For the implementation of these functions, [TS00] has been used as a reference guide.

Files: sound.c and sound.h

Functions: `setup_mixer()`, `change_mixer_level()`, `setup_audio()`, `fill_audio_buffer()` and `play_audio()`

D.4.1 setup_mixer()

Function to setup the mixer device.

Input: debug – flag to tell whether to output anything or be silent

Output: File descriptor or Failure.
File descriptor for the mixer if it was set up with the defined volume.
FAILURE if something went wrong.

Errors: all errors are reported as returned FAILURE.

The setup of the mixer is rather simple, using the OSS API. The device file */dev/mixer* is opened, and an attempt to write a predefined volume between 1 and 100 is made.

D.4.2 change_mixer_level()

Function to change the level of the mixer device.

Input: File descriptor and level.
mixer_fd – file descriptor for the mixer device level – the level at which the mixer should be set.

Output: Success or Failure.
Success if the mixer was set up with the defined volume.
FAILURE if something went wrong.

Errors: all errors are reported as returned FAILURE.

Changing the mixer level is done by attempting to write the specified volume between 1 and 100.

D.4.3 setup_audio()

Function to setup the audio device.

Input: debug – flag to tell whether to output anything or be silent.

Output: File descriptor or Failure.
File descriptor for the audio device if it was set up properly.
FAILURE if something went wrong during setup.

Errors: all errors are reported as returned FAILURE

Setting up the audio device requires slightly more than setting up the mixer. The routine is to first open the audio device */dev/dsp* and then setup the audio format, number of channels and sample rate.

D.4.4 fill_audio_buffer()

A function to chop a buffer of 16 bit values into chars

Input: The input and output buffers.
wav_buffer – pointer to a signed 16 bit buffer.
audio_buffer - pointer to unsigned 8 bit buffer

Output: Success or Failure.
SUCCESS if the conversion went ok.
FAILURE if something went wrong.

The function works by using bit shift and binary AND, to copy the low and high bytes of each short in the input buffer to the output buffer.

D.4.5 play_audio()

Function to play audio from a buffer.

Input: File descriptor and buffer.
audio_fd – file descriptor for the audio device.
audio_buffer - pointer to unsigned 8 bit buffer.

Output: none

Errors: exit program with EXIT_FAILURE if we cannot write to the device.

Playing audio is done by feeding the interleaved 16 bit stereo samples in the char buffer to the audio device.

D.5 WAVE

A set of functions for reading, checking and processing WAVE files. Only the functions in this category are used for tasks related to WAVE files and the data from them.

Files: wav_functions.c and wav_functions.h

Functions: `check_wav_file()`, `read_wav_head()`, `chkheadstr()`, `load_wav_buffer()`, `process_wav_buffer()` and `get_file_ints()`

Type definitions: `struct wavheader`

D.5.1 check_wav_file()

A function to check whether the WAVE file fulfills the requirements of the system.

Input: filename – string (char array/pointer) with the filename to be checked

Output: the result of the file check.
WAV_FILE_OK if the file can be read and has a proper format.
WAV_FILE_BAD if there is something wrong with the file.

The `check_wave_file()` function first opens the given WAVE file and loads the header, using `read_wav_head()`. If the file is not a WAVE file, the header cannot be read, and this message is written in the header information struct, which will make the function return WAV_FILE_BAD. If the file is a WAVE file, it is controlled whether it fullfills the requirements of the system with regard to sample rate and the number of channels and bits per sample and the result is then returned.

D.5.2 read_wav_head()

A function to read the header of a WAVE file into a struct.

Input: File pointer and filename.
*ifp – a file pointer to the WAVE file.
filename - string (char array/pointer) with the filename.

Output: struct of the type wavheader with WAVE info, including number of channels, samplerate, bitrate, sample size and more.

Errors: if the header is not a valid WAVE header, the legal flag of the struct is set to WAV_FILE_BAD and an unfinished struct is returned.

Reading the header of a WAVE file is don by reading one header field at a time, where the fields are defined in [IM91]. If any of the fields indicate, that the file is not a WAVE file, then the function returns with the WAV_FILE_BAD flag. Two sub functions are used by this function: `chkheadstr()` which checks whether the next characters of the file are equal to the specified string, and `get_file_ints()` which reads an integer of a given number of bytes from the file.

D.5.3 chkheadstr()

Check the current header for matching a specific string. This function is solely used by the `read_wav_head` function because a number of four character strings must exist in a WAVE file header.

Input: File pointer, filename and the header string to compare to.
*ifp - file pointer to the WAVE file.
filename - string (char array/pointer) with the filename.
header - string (char array/pointer) with the control string.

Output: True or False.
TRUE if the header matches.
FALSE if the header does not match or end of file is reached.

The function works by reading four bytes from the file and comparing them to the string of the input.

D.5.4 load_wav_buffer()

Load part of the WAVE data into a 16 bit buffer.

Input: File pointer, header information, buffer and the next sample number to process.
*wav_fp – file pointer to the WAVE file.
wavhead – struct with WAVE header information.
buffer – pointer to the buffer to fill.
sample_num - The next sample number of the WAVE file to be read.

Output: sample_num - The first sample number to read next time.

This file fills up the input buffer the number of WAVE file samples that are defined in global.h. The samples will be converted 16 bit signed stereo samples if they are not already, and if the end of the file is reached, the remaining part of the buffer will be filled with zeros.

D.5.5 process_wav_buffer()

Perform calculations on the WAVE buffer to adjust the gain.

Input: buffer and settings.
buffer - pointer to the buffer to do processing on.
settings - a struct with settings to control the processing.

Output: none

In the settings struct a floating point gain is saved, which is multiplied with the signal.

D.5.6 get_file_ints()

Read an integer of ”size” bytes from a file. This function is used to read a proper size variable from a WAVE file, depending on the bit size.

Input: File pointer and size.
*ifp – File pointer to the file.
size – Number of bytes that should be converted into an integer.

Output: The value which was read from the file.

Errors: Will exit the program on file read error.

This function attempts to read a given number of bytes (maximum 4) into an integer, and takes care of error handling. The function is only used as a subfunction for the other WAVE functions.

D.5.7 wavheader struct

The struct containing the WAVE file header information has the following definition.

```
typedef struct wh {
    char *filename;   // Name of the WAVE file
    int filesize;     // Size of the file, according to the header
    int samplerate;   // The sample rate in Hz
    int num_chan;     // Number of channels interleaved
    int byterate;     // Bytes per second
    int block_align; // Number of bytes per sample
    int bits;         // Bits per sample
    int datasize;     // Size of the WAVE data (without the header)
    int num_samples; // Number of samples
    int legal;        // Flag to report if something is wrong with the WAVE file
} wavheader;
```

D.6 Settings

Functions for setting and changing the playback settings. Settings are transferred to a running instance by writing the settings struct to a binary file, which is then read by the running instance.

Files: settings.c, settings.h

Functions: `write_settings()`, `read_settings()` and `lock_settings()`

Type definitions: `struct sett_struct`

D.6.1 write_settings()

Write the settings to a binary file. While writing, the file is locked, so a running process will not attempt to read it. If the file is already locked, the process will wait until it is available and then write.

Input: Settings.
settings – A struct of the type defined in settings.h.

Output: Success or Failure.
SUCCESS if settings were sucessfully written to the file.
FAILURE if it was not possible to write the settings to the file.

This function first tries to open and lock the settings file. If the file is locked (someone else is reading or writing), then the function waits for it to be available. When the file is available, the settings are written in a binary format by simply dumping the settings struct to the file, and finally releasing the file again.

D.6.2 read_settings()

Read the settings from a binary file. Before attempting to read, make sure the file is not locked. If any error occurs, the old settings are returned intead of new ones.

Input: The currently used settings.
old_settings – A struct of the type defined in settings.h

Output: The settings to use next.
settings – A struct of the type defined in settings.h

Errors: upon failure to read, the old settings are returned instead of new ones

Reading the settings file is done by first trying to open and lock the file. Unlike when writing, if the file is locked, the function will not wait, but instead return the already used settings, since this function will be called in the part of the program, which has timing constraints due to audio being loaded and played. If the file is not locked, the file will be read by copying the binary format directly back to a setttings struct, which will the be returned.

D.6.3 lock_settings()

Open the settings file and try to lock it. This function is used by the write and read settings functions to attempt to open and lock the settings file.

Input: None.

Output: File descriptor of the opened file or error

Errors: Failure or the settings are locked.
FAILURE if the file cannot be opened.
SETTINGS_LOCKED if there is a lock on the file

The function works by first attempting to open or create the settings file and then test if it is locked. If the file is locked, then it is closed and SETTINGS_LOCKED is returned. If the file is not locked, it's file descriptor will be returned.

D.6.4 struct sett_struct

The struct containing the settings is defined as follows:

```
typedef struct sett {
    char filename[MAX_NAME_LEN]; // Filename of the WAVE file to play - must be a fixed
       length to write to binary file easily
    float gain;                  // The gain setting as a floating point number
    int mixer_level;             // Level of the OSS mixer.
    int num_plays;               // Number of times to repeat the WAVE file
    int debug;                   // Debug flag to control various aspects of the program
    int changed;                 // Flag indicating whether new settings are in use
} sett_struct;
```