

Low Power Sensor Network for Collective Hearing

Semester Project

10. February 2005

Author : Vlad Trifa

Supervisor : Christopher Cianci

Professor : Alcherio Martinoli

Contents

| | |
|--|------------|
| Abstract | vii |
| 1 Introduction | 1 |
| 2 Sound Basics | 3 |
| 2.1 On waves | 3 |
| 2.2 Sound propagation in air | 4 |
| 2.2.1 Geometric spreading | 4 |
| 2.2.2 Atmospheric Effects | 5 |
| 2.2.3 Surface Effects | 6 |
| 3 Sensor Networks | 9 |
| 3.1 Wireless sensor networks | 9 |
| 3.1.1 Design Constraints | 10 |
| 3.2 Hardware Design | 11 |
| 3.2.1 MicaZ Motes | 11 |
| 3.2.2 MTS300CA Sensor Board | 11 |
| 3.3 Tiny Operating System (TinyOS) | 12 |
| 3.4 Component Types | 13 |
| 3.4.1 Components | 13 |
| 3.5 TOSSIM, a TinyOS simulator | 15 |
| 3.6 Power Management | 15 |
| 3.7 Time Synchronization | 18 |
| 3.7.1 In-Network Processing | 18 |
| 3.8 Sound Detection | 19 |
| 3.8.1 Frequency limitations in sensor networks | 20 |
| 3.9 Implementation | 21 |
| 3.9.1 Mote Algorithm | 21 |
| 3.9.2 Central computer | 21 |
| 4 Sound Perception | 25 |
| 4.1 Human Auditory System | 25 |
| 4.2 Human and Animal Sound Localization | 27 |
| 4.3 Sound localization in biology | 29 |
| 4.3.1 Binaural neurons sensitivity | 30 |
| 4.3.2 3D Localization | 30 |

| | | |
|----------|---|-----------|
| 5 | Localization and detection | 33 |
| 5.1 | Sound localization | 33 |
| 5.1.1 | Localization techniques | 34 |
| 5.2 | The Situation | 35 |
| 5.3 | Mathematical Model | 36 |
| 5.4 | Fourier Basics | 37 |
| 5.4.1 | Discrete Time Fourier Transform (DTFT) | 37 |
| 5.4.2 | Discrete Fourier Transform (DFT) | 38 |
| 5.4.3 | Short-Time Fourier Transform (STFT) | 38 |
| 5.5 | Localization Techniques | 39 |
| 5.5.1 | Stereausis Approach | 41 |
| 5.6 | Sensor fusion | 41 |
| 5.6.1 | Beamforming | 42 |
| 5.7 | Implementation | 42 |
| 5.7.1 | Determination of Minimal Sampling Frequency | 42 |
| 6 | Results and discussions | 45 |
| 6.1 | Experiments | 45 |
| 7 | Conclusion and Future Work | 49 |
| 7.1 | How to use TOS | 51 |
| 7.1.1 | Compile a program | 51 |
| 7.1.2 | Data retrieval | 51 |
| 7.2 | The Java Localizer Program | 53 |
| 7.2.1 | Sound Files | 53 |
| 7.3 | The Matlab GUI | 55 |
| 7.3.1 | Environments | 57 |
| | Bibliography | 57 |

List of Figures

| | | |
|-----|---|----|
| 2.1 | Attenuation of sound energy in dB/100m in function of relative humidity and temperature of air | 7 |
| 3.1 | Current draw for each device on the MICAZ mote | 16 |
| 3.2 | The finite state machine describing the controller of the program running on the mote. | 22 |
| 3.3 | Algorithm running on the motes | 22 |
| 4.1 | Human auditory system. | 26 |
| 4.2 | Tonotopicity of the basilar membrane. | 27 |
| 4.3 | Hair cells of the basilar membrane. | 28 |
| 5.1 | Geometric representation of the system composed by binaural sensors and the sound source. | 35 |
| 5.2 | The structure of the stereausis network | 41 |
| 5.3 | The value of the time delay change for a variation of 1 for different values of the angle of arrival θ | 43 |
| 6.1 | The average packetloss percentage for multiple (from 1 to 9) motes emitting in the same time, at different sampling frequencies. | 46 |
| 6.2 | The location estimate by intersection of the bearing yielded by two pairs of motes. The blue areas are the cone of confusion of each pair. . | 47 |
| 7.1 | Interface of the java localizer program used to transform the packets from the network into Matlab scripts containing the sensor readings correlated in time. Long diagonals denote packetloss. | 54 |
| 7.2 | Example of a sound file generated by the Localizer program. | 54 |
| 7.3 | Graphical interface of the sound localization library. | 55 |
| 7.4 | Binaural signals. | 56 |
| 7.5 | General command for the sound signals processing and pair selection. | 56 |
| 7.6 | Command panel for filtering and localization of sounds. | 57 |
| 7.7 | Panels containing information concerning binaural signals. | 57 |
| 7.8 | Example of the content of the configuration file for a given environment. | 58 |

Abstract

This project presents an implementation of a robust and energy-efficient localization algorithm based on information retrieval using a sensor network. The sensor nodes are controlled by an efficient and compact program defined by a finite state machine whose task is to sample the microphone readings and send the data back to a central computer as soon as a sound is detected. The role of the central computer is to transform the data into Matlab scripts for later processing. We present the Matlab Sound Localization Toolbox, which is a graphical tool inspired by the modular architecture of the Auditory Toolbox developed by Malcom Slaney. This Matlab package is a graphical tool for performing sound localization and is able to plot the auditory scene. It is a very modular and evolvable framework as additional filters and localization techniques are easily added. A description of the biological sound localization processes and the structures it involves is presented and an overview of some common mathematical localization methods widely used nowadays is also given. A final discussion towards the future work and evolution of sensors nodes is done.

Chapter 1

Introduction

The ability to accurately estimate the location of a sound source has obvious evolutionary advantages in terms of locating prey and avoiding predators. Evidence of that statement is given by the broad range of techniques used by animals for determining the direction of a sound. Bats for example, perceive the world solely using their ears for listening the returning echoes of short ultrasonic pulses they emit, so they need a fine tuning of their sound processing apparatus in order to deal efficiently with unpredictable and dynamic environments. Owls cannot use their visual system to locate a prey during the night and thus need to use other sensory systems for hunting. There has been a considerable amount of psychoacoustical research in understanding how does the sound localization process take place in humans and animals, and many various models have been proposed. Nonetheless, there still remains a large gap between psychophysical experiments and the explanations yielded by such more or less mathematical models. At the present time, no one of these computational methods is able to deal efficiently with a wide-range of sounds (varying in bandwidth and center-frequency, spectrally complex sounds, robustness to irregularity, etc) with an efficiency comparable to the one reached by the human auditory system. Furthermore, the biological constraints concerning sound localization have generally not been explored by these models, but rather an engineering approach has systematically been taken. These constraints include the spectral resolution of the auditory system in terms of the number and bandwidth of frequency channels and the role of the tonotopic¹ processing.

In this project we developed a generic framework under Matlab devoted to perform multisensor location analysis using sound wave processing, and a GUI that allows the user to easily interact with the software. This platform is rather devoted to scientific research than to an industrial application, as there does not yet exist to our knowledge such a framework specific sound localization specific to sensor networks. The architecture is very modular, and new functionalities are easily added and integrated, assuming that the system specifications are respected. Then a graphical environment we developed allows easy plotting of the environment and of the elements within, as sensors, sound source and the estimated location. This is still an experimental able to analyze offline the `log` files generated by another application which retrieves the sensor data from the network and creates a matlab script containing all the data concerning an isolated auditory event.

¹refers to a spatial map in the auditory cortex where each area is associated with a specific frequency, by being innervated by the frequency dependent nerve fibers coming from the cochlea

The outline of this document is as follows :

Chapter 2 reviews important facts about sound propagation and how environmental conditions can influence the speed of sound.

Chapter 3 describes the hardware and software tools we used. This chapter also introduces the most important facts we need to take into account when developing an efficient sound localization algorithm such as power consumption, time synchronization, sampling frequency, etc.

Chapter 4 describes the organization of the human and animal auditory systems and how they perceive and process sounds. Then we describe how to draw inspiration from the natural processes in order to develop a neuromorphic sensor board for MicaZ nodes dedicated to sound processing and localization.

Chapter 5 introduces to the situation and describes the main methods commonly used in sound localization and describes the method we implemented in our Matlab Toolbox. We also explain how to use the package we created in order to process sound.

Chapter 6 shows the experiments we carried out and explains theoretical facts about accuracy we get, what factors influence the minimal accuracy attainable with our system.

Chapter 7 concludes this project and explains the further work and possible directions in order to improve the accuracy of the sound localization.

Chapter 2

Sound Basics

2.1 On waves

Definition Sound is a mechanical wave that results from the longitudinal motion of the particles of the medium through which the sound wave is being propagated.

Let's take for example a vibrating tuning fork. The sound it produces is heard due to the tines of the fork vibrating back and forth pushing on neighboring air particles. When a tine pushes air molecules horizontally thus compressing them next to the direction of vibration, the quick backward retraction of the tine will create a low-pressure area allowing the air particles to move back to the left. Because of the longitudinal motion of the air particles alimented by the energy of the sound wave propagation, there are regions in the air where the air particles are compressed together (*compressions*) and other regions where the air particles are spread apart (*rarefactions*).

Since a sound wave consists of a repeating pattern of high pressure and low-pressure regions moving through a medium, it is often referred to as a pressure wave. A microphone is simply a device that is sensitive to small and quick fluctuations in pressure as the sound wave impinges its membrane. At one instant in time, the detector would detect a high pressure; this would correspond to the arrival of a compression at the detector site. Since the fluctuations in pressure as detected by the detector occur at periodic and regular time intervals, a plot of pressure vs. time would appear as a sine curve. The crests of the sine curve correspond to compressions; the troughs correspond to rarefactions; and the "zero point" corresponds to the normal pressure of the air into the environment when no sound is present. Yet, the representation of sound waves as sinusoidal waves might be somewhat misleading if one is not careful. This representation is just an attempt to illustrate the sinusoidal nature of the pressure-time fluctuations. One should not conclude that sound is a transverse wave which has crests and troughs. Sound is indeed a longitudinal wave with compressions and rarefactions. As sound passes through a medium, the particles of that medium do not vibrate in a transverse manner but in the direction of energy transport and this is what characterizes sound as a longitudinal wave.

Since a wave repeats its pattern once every wave cycle, the wavelength is sometimes referred to as the length of the repeating pattern of pressure variations. For a transverse wave, this length is commonly measured from one wave crest to the next

adjacent wave crest. Since a longitudinal wave does not contain crests and troughs, its wavelength must be measured differently. Thus, the wavelength is commonly measured as the distance from one compression to the next adjacent compression. Of course, the frequency can be directly derived as it is the inverse of the wavelength.

The ears of humans are sensitive detectors capable of perceiving the fluctuations in air pressure that impinge upon the eardrum. It is interesting to note that the human ear is capable of detecting sound waves in a wide range of frequencies, ranging between approximately 20 Hz to 20 kHz. Some animals possess the ability to detect sounds that are not audible for a human subject as dogs for example, are able to detect ultrasounds up to a frequency of 45 kHz, and cats are able to reach 85 kHz. Bats, who are essentially blind and must rely on sound echolocation for navigation and hunting, can detect frequencies as high as 120 000 Hz. Dolphins can detect frequencies as high as 200 000 Hz. While dogs, cats, bats, and dolphins have an unusual ability to detect ultrasound, elephants on the contrary possess the unusual ability to detect infrasound, having an audible range from approximately 5 Hz to approximately 10 000 Hz. Some studies demonstrated that they are even able to communicate by emitting low-frequency waves by trampling the ground with their huge feet.

The sensations of these frequencies are commonly referred to as the pitch of a sound. A high pitch sound corresponds to a high frequency and a low pitch sound corresponds to a low frequency. Interestingly, some musicians are capable of detecting a difference in frequency between two separate sounds that is as little as 1 Hz. When two sounds with a frequency difference of greater than 7 Hz are played simultaneously, most people are capable of detecting the presence of a complex wave pattern resulting from the interference and superposition of the two sound waves.

2.2 Sound propagation in air

It's important to take into account the fact that the speed of sound is influenced by environmental factors such as temperature and relative humidity in air. These important factors are classified into 3 main categories : geometric spreading, atmospheric effect and surface effects. At small-scale experiments done in a laboratory not all of these effects are to be taken into consideration and the importance of these various phenomena depends upon the situation under consideration.

2.2.1 Geometric spreading

This refers to the spreading of sound energy as a result of the expansion of the wave fronts. There are two common kinds of geometric spreading: spherical and cylindrical spreading. Sound propagation losses due to spreading are normally expressed in terms of dB per doubling of distance from the source.

An impulse applies a chunk of energy to the air. (This energy becomes a difference in pressure before and behind the wave front.) As the front expands, the energy is spread over a larger and larger area, in a way suggested by the relationship between the area A of a sphere and its radius R :

$$A = 4\pi R^2 \tag{2.1}$$

The total energy remains constant while the area expands, so the energy in one unit of area decreases with the square of the distance from the source. The front

will continue moving until there isn't enough energy to measure. The stronger the impulse, the farther the wave front goes (the sound of the explosion of the volcano Krakatoa is said to have gone around the world three or four times). This is called the *inverse square law*. In the case of spherical spreading from a point source, which is due to a noise source radiating sound equally in all directions, the sound level is reduced by 6 dB for each doubling of distance from the source, while a line source will produce cylindrical spreading, resulting in a sound level reduction of 3 dB per doubling of distance.

In some situations, sound may actually propagate much farther than implied by the inverse square law. That happens when the energy is somehow focused rather than evenly distributed on an expanding sphere. In fact, sound propagation is quite rarely homogenous and spherical. Some examples of this "focusing effect" :

Directional loudspeakers We can use a speaker that makes the sound propagate mostly in one direction. In fact, it would be rather difficult to build a real omnidirectional. Directionality is always more pronounced at high frequency than low.

Reflective support The back wall of a bandshell and the ceiling of a stage are designed to bounce sound into the audience. The mountains do the same for train horns.

Atmospheric effects Since the speed of sound differs according to air temperature, sound traveling along a boundary between warm and cold air tends to get bent, and that will make the sound being diverted into the cold region. We call this phenomenon *refraction* because it's similar to light when it changes medium of propagation.

2.2.2 Atmospheric Effects

Air Absorption

There are two mechanisms by which acoustic energy is absorbed by the atmosphere. They are called *molecular relaxation* and *viscosity effects*. Molecular relaxation is by far the most important.

The expression of the speed of sound in the air at 0 °C is c_0 :

$$c_0 = \sqrt{\kappa \frac{p_0}{\rho_0}} \quad (2.2)$$

where p_0 is the atmospheric air pressure equal to 101'325 Pa in standard conditions (0 °C), ρ_0 is the density of air given by the gas law (in standard conditions equals to 1.2923 kg/m³), κ is the adiabatic exponent of air at 0 °C. That is the ratio of its specific heat at constant pressure c_p to its specific heat at constant volume c_v :

$$\kappa = \frac{c_p}{c_v} \quad (2.3)$$

One can easily understand the relation if one thinks of the following experiment: A cylinder closed at one end with a blocked piston is filled with air. This cylinder is then heated. Since the piston cannot move the volume is constant. Temperature and pressure will rise. One stops the heating noting how much energy one has added to the system (c_v) and frees the piston. The piston will move outward and by adiabatic

expansion the gas cools. To bring the temperature of the gas to its former level one has to heat it a little bit. This extra heat amounts to about 40% of c_V thus:

$$c_p = \kappa \cdot c_V = 1.402 \cdot c_V \quad (2.4)$$

That brings us to find the speed of sound in normal conditions :

$$c_0 = \sqrt{\frac{101'325}{1.2935} \cdot 1.402} = 331.4\text{m/s} \quad (2.5)$$

We can generalize these results and get the formula for speed of sound at temperature ϑ (in °C).

$$c_{\vartheta} = c_0 \cdot \sqrt{1 + \alpha \cdot \vartheta} \quad (2.6)$$

where α is called the *coefficient of expansion* which is $1/273.15 = 3.66 \cdot 10^{-3}$ expressed in $[C^{-1}]$ Another interesting aspect of equation 2.2 is that the air density ρ can be expressed as a function of the temperature T and gas constant R which equals 287.05 J/kg · K for dry air.

$$\frac{p}{RT} = \frac{n}{V} = \rho \quad (2.7)$$

After replacing this last equation in 2.2, we get

$$c_0 = \sqrt{\kappa RT} \quad (2.8)$$

we may immediately notice that the fraction p_0/ρ_0 is always constant, which means that the air pressure and air density are proportional to each other at the same temperature. This implies that the speed of sound does not depend on the air pressure, but on temperature. Humidity also has a small influence because the amount of vapor in air directly influences the gas constant R , because it varies according to air density and the moister is the air the less it is dense. Moisture also causes the specific-heat ratio to decrease, which would cause the speed of sound to decrease. However, the decrease in density dominates, so the speed of sound increases with increasing moisture.

Sound engineers usually use a simplified version of this formula with sufficient accuracy for general applications :

$$c_0 = 331.4 + 0.6 \cdot \vartheta \quad (2.9)$$

This equation is interesting for an approximation of the speed of sound, as it can be easily implemented on the restricted hardware of the notes.

2.2.3 Surface Effects

Ground Absorption

A sound propagating over ground will be attenuated due to energy losses on reflection, which depend on nature of the reflective surface. Smooth and hard surfaces will produce little absorption whereas thick grass may result in sound levels being reduced by up to about 10 dB per 100 meters at 2000 Hz. High frequencies are generally

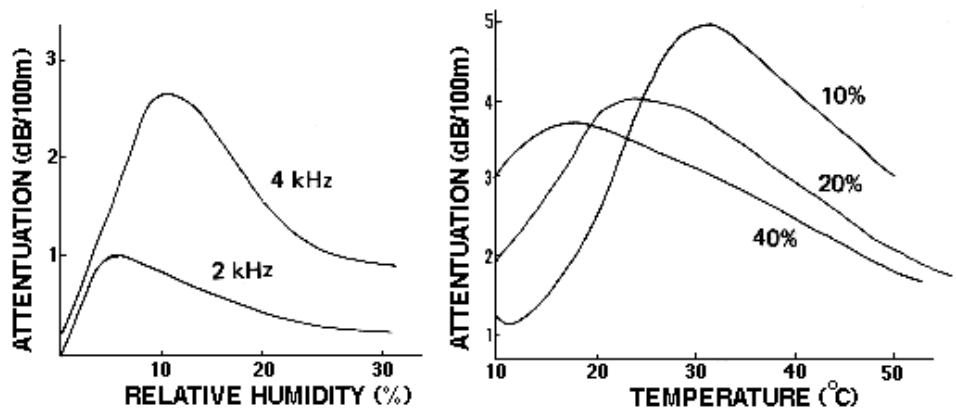


Figure 2.1: Attenuation of sound energy in dB/100m in function of relative humidity and temperature of air

attenuated more than low frequencies, and for that reason we hear only the low frequency components of the music produced by a concert as we move far away.

Reflection from the ground can result in another mechanism by which sound levels are reduced by destructive interferences from the reflected sound when both the receiver and the emitter are close to the ground. This is called the *ground effect* and is normally noticed over distances of several meters and more, and in the frequency range of 200-600 Hz.

Attenuation Due to Barriers

Solid barriers can achieve significant attenuation and can modify the sound wave propagation depending on their physical properties. High frequency waves are more attenuated since low frequencies tend more to diffract around the edge of an obstacle. Such obstacles attenuate the most when placed very close either to emitter or the receiver. Also note that the amount of attenuation might vary according to temperature and wind gradients.

Chapter 3

Sensor Networks

3.1 Wireless sensor networks

A recent and trendy topic in research as well as a very promising industrial gold mine, is what we call *Smart Environments*. They represent the next step towards evolution in buildings, industrial plants, home, laboratories or ecosystems, for performing a large panel of different tasks ranging from gas leaks monitoring to intrusion detection. The smart environment relies on the sensory data coming from the environment collected through a large scale, high density, dynamic and data centric network composed of a set of spatially distributed tiny and autonomous devices networked by wireless communication usually called *motes* or *nodes*. The data can then either be distributed on multiple motes for being processed, or it can also be directly sent to a computer connected to this network through a mote attached to the serial port. These platforms are still a topic of active research in a number of different communities in industry and academia, ranging from hardware to applications. The whole system has a global internal state and is able to perform a collective decision-making process in function of the collected data and then act upon the environment in question through dedicated activators. This set of interconnected "organs" can lead to the interesting analogy with a living organism, in that for proper functioning, the system has to deal efficiently with both *proprioceptive*¹ as well as *exteroceptive*² information. *Wireless Sensor Networks (WSN)* are characterized by a need to carefully integrate functionalities initially from separate domains in order to achieve maximum efficiency, especially with respect to energy consumption and data management. Hence, a close interaction of research from different technical backgrounds is required. Since its early days WSNs have been constantly evolving from simple data transportation networks towards functionally rich distributed and autonomous systems.

Some of the common applications envisioned for sensor networks range from monitoring of habitats for animals, detection of contaminants and pollutants in fluids, gas leaks, chemical plants, vehicular traffic monitoring, floods, forest fires, earthquakes, tornadoes, tsunamis, volcanoes, national and personal security, detection of intrusion and tracking of targets, etc. Unfortunately, most of the sensor network applications need to perform large volumes of data processing, and cannot yet be efficiently implemented on sensor nodes. The detection/analysis of events are based on the integration

¹The ability to sense it's own internal state, activators in machines, muscles, organs in animals.

²The ability to sense external state, through sensory stimuli.

of multiple sensory information (audio, video, thermal, CO₂ concentration, etc.) at different spatial locations.

3.1.1 Design Constraints

In such systems, we must carefully respect some essential constraints in order to ensure proper functioning of the system, as stated in [HSW⁺00] are :

Minimal required resources These devices are intended to be deployed in mass and hence they must be small, in order to minimize the power consumption and dissipation. The OS running on these devices must also efficiently use the hardware in order to require the minimal amount of energy for performing its duty. All these constraints limit the amount of processing, storage and communication, thus they're driving factors in the hardware design and must be carefully studied according to the task our sensors have to deal with.

Concurrency-intensive operation The primary mode of operation for these devices is to forward data from place to place with a modest amount of processing on the fly, rather than analyze each packet and then decide what to do with it. The problem that could arise in this situation, is when data must be captured from sensors and streamed onto a network while the mote still needs to forward data coming from other motes, thus acting as a router. As there is little internal storage capacity, buffering large amounts of data is not particularly feasible.

Limited Physical Parallelism and Controller Hierarchy Due to their limited physical size, the number of independent controllers and their capabilities are much lower than in conventional systems. Typically, the sensor or actuator provides a primitive interface directly to a single-chip microcontroller. In contrast, conventional systems distribute the concurrent processing associated with the collection of devices over multiple levels of controllers interconnected by an elaborate bus structure. Space and power constraints and limited physical configurability on-chip are likely to drive the need to support concurrency-intensive management of flows through the embedded microprocessor.

Diversity in Design and Usage Networked sensor tend to be application specific rather than general purpose and carry only the necessary hardware needed for the application of interest in order to avoid extra energy expenses by unused elements. As there is a wide range of potential applications, the variation in physical devices is likely to be large. On any particular device, it is important to easily assemble just the software components required to synthesize the application from the hardware components. Thus, these devices require an unusual degree of software modularity that must also be very efficient. A generic development environment is needed which allows specialized applications to be constructed from a spectrum of devices without heavyweight interfaces.

Robust Operation These devices will be numerous and largely unattended, thus we need to create an application which will be operational a large percentage of the time. Depending on their use, failures could be dangerous and might even cause damages or injuries. Traditional redundancy techniques used to enhance the reliability of

individual units are limited by space and power and even if redundancy across devices seems attractive, the communication cost of such a method is too prohibitive. For that reason the goal is to enhance the reliability of individual devices in order to ensure robustness at the network level.

3.2 Hardware Design

3.2.1 MicaZ Motes

The hardware we used is the Crossbow MICAZ motes, developed at UC Berkeley and commercialized by the XBOW company. It consists of a real computer microcontroller with internal flash program memory, data SRAM and data EEPROM, connected to a set of actuator and sensor devices, including LEDs, a radio transceiver, an analog photo-sensor, a microphone, a 4kHz buzzer, a digital temperature sensor, a serial port and a small coprocessor unit.

The processor within the MICAZ mote (ATMEL ATmega128L), it is an 8-bit Harvard architecture with 16-bit addresses. It provides 32 8-bit general registers and runs at 8 (7.37) MHz at 3.0 V. The system is very memory constrained: it has 128 Kb of flash as program memory, and 512 Kb of SRAM as data memory. The processor also integrates a set of timers and counters that can be configured to generate interrupts to the processor at regular time intervals. Three LEDs may be used to display digital values or status. The photo-sensor represents an analog input device with simple control lines. In this case, the control lines eliminate power drain through the photo resistor when not in use. The input signal can be directed to an internal ADC in continuous or sampled modes. The radio is the most important component as it represents an asynchronous input/output device with hard real time constraints. It consists of an IEEE 802.15.4/ZigBee compliant RF receiver, which allows a data rate of 250 kbps by emitting in the 2.4 GHz band. Secured transmissions can be handled via a protocol called Tinysec, which is similar to WEP in 802.11g networks.

3.2.2 MTS300CA Sensor Board

This is the sensor board we used for this project. It has a light sensor, a thermistor, a sounder and a microphone. The basic circuit consists of a pre-amplifier (U1A-1), second-stage amplified with a digital pot-control (U1A, PT2). This circuit amplifies the low-level microphone output, which can be fed directly into the analog-digital converter (ADC2) by using the Microphone Output selector circuit (MX1) to connect the mic_out signal to ADC2 signal. This configuration is useful for general acoustic recording and measurement. The second stage output is routed through an active filter into a tone detector that turns the analog microphone signal into a digital high or low level output when a 4 kHz output is present. The sounder generates this 4 kHz tone, and this latter configuration is used for acoustic ranging by pulsing sounder and sending an RF packet at the same time. The other motes will then determine approximately their distance from other motes by calculating the *Time-of-Flight* of the sound wave (as we assume that the RF packet is received simultaneously compared to the sound wave).

Signals coming from sensors can be very noisy, of low amplitude, biased and dependent on many other factors such as environmental conditions. Moreover, we cannot always be able to measure the absolute quantity of interest without a precise calibra-

tion, but generally only the relative values. For that reason one might want to perform *Signal Conditioning* or process the signal to extract the meaningful information. A simple technique for improving the *Signal-to-noise ratio* or SNR is low-pass filtering since noise is usually present in high frequencies. This could be done directly using hardware (simple RC circuit) or in software, either on the mote or on the central PC.

3.3 Tiny Operating System (TinyOS)

The tiny embedded devices we described in the previous section need a small yet powerful OS fitting on the limited hardware resources of the MICAZ motes. For that purpose, the motes are running TinyOS, an open-source operating system especially designed for wireless sensor networks. TOS uses a component-based architecture, which allows a radical minimization of application code as required by the severe memory constraints these systems suffer. This system architecture model makes the system much more modular and flexible, allows quick development of the application as each "functionality" is implemented inside a generic module (the component). A library of ready-made components is integrated into TinyOS, which contains network protocols, distributed services, sensor drivers, data acquisition tools, etc. which can be easily integrated into any application in a similar way one includes programming libraries .h headers in C programs. The main difference resides in the fact that components in TinyOS are bidirectional interfaces. That implies in their use, not only to be able to call functions from that library, but also implement functions that the library itself might call within your application. To be more precise, this second class of functions, are called *events* and you have to react appropriately to these events by implementing the code to run when the event is signaled. For example, when you use the RF component, you are able to call the function `sendMessage(message)`. But upon arrival of a message from the network, the same interface requires you to implement the event `messageReceived(message)`, where you unpack this message if necessary and process the data it contains as you want. A complete system, is composed of two schedulers and a graph of components within the application. The tasks which perform the main task are atomic to each other, so they are just processed in FIFO order, though they can be preempted by events generated by timers or in response to hardware interrupts from sensors or radio. Once all tasks in the queue have been served, there is no reason to keep the processor running and it should be turned into idle state until a new event arrives and post new tasks. Unfortunately, this simple scheduler is not able to meet the requirements needed by hard real-time constraints and thus is prone to failure in time-critical applications. For this purpose, more sophisticated schedulers using priority levels or deadline monotonic scheduling method can replace the original one. TOS event-based execution model enables a fine-grained power management, while still providing a sufficient flexibility needed by the unpredictable nature of the interaction with the physical world (continuous time, frequent variations in the perceived environmental pattern, noise, etc.) This event-based model is in fact the most efficient programming paradigm that is able to use the CPU power efficiently, by adjusting autonomously its parameters according to the external situation. This approach eliminates continuous polling and the related energy expenses and allows us to efficiently handle asynchronous time events.

Another very practical feature in TinyOS, is the *Active Message* paradigm for message-based communication. It is a simple and extensible model widely used in parallel and distributing systems. Each message has a specific a number that defines

its type and associates this type to a specific handler to invoke by the networking layer upon arrival of each message of this type. In case the handler is not implemented, that means the mote does not deal with this kind of message and the packet will be discarded (or forwarded if in a multihop network) without being opened in order to analyze the content. This method allows to avoid energy expenses induced by having each packet opened by the application as the handler is invoked only if low-level layers are programmed to forward this type of packet to higher levels.

3.4 Component Types

In general, components fall into one of three categories: hardware abstractions, synthetic hardware, and high-level software components.

Hardware Abstraction Components These components simply map physical hardware into our component model, for example the RFM radio component. This allows us to program our motes by direct interaction with the hardware through low-level procedures. This component exports commands to manipulate the individual I/O pins connected to the RFM transceiver and posts events informing other components about the transmission and reception of bits. Its frame contains information about the current state of the component (the transceiver is in sending or receiving mode, the current bit rate, etc.). This model of abstracting over the hardware resources can scale from very simple resources, like individual I/O pins, to quite complex ones, like UARTs.

Synthetic Hardware Components They simulate the behavior of advanced hardware. A good example of such component is the **RadioByte** component, which shifts data into or out of the underlying **RFMmodule** and signals when an entire byte has completed. The internal tasks perform simple encoding and decoding of the data. From the point of view of the higher levels, this component provides an interface and functionality very similar to the UART hardware abstraction component: they provide the same commands and signal the same events, deal with data of the same granularity, and internally perform similar tasks (looking for a start bit or symbol, perform simple encoding, etc.).

High Level Software These are all other components. They do generally provide a higher-level interface compared to the two previous categories. They implement higher-level task such as control, routing and all data transformations through calculation and aggregation. In fact the applications one programs for sensor networks generally fall into this category.

3.4.1 Components

The extreme modularity achieved by the TinyOS scheme, has allowed the development of many different general-purpose and application-specific libraries, called modules, which could be easily integrated in any application simply by binding together the needed interfaces.

XNP This module is also called in-network reprogramming. It allows by adding some lines in any of your application to be able to reprogram the code stored on a mote directly by sending it the new program in chunks of data called capsules directly by radio, without need to physically attach the mote to a base-station. When all capsules are sent, the motes with missing capsules will request them, and when all capsules are received, they'll reprogram themselves by loading the new code into their memory and reboot using the new software. This allows autonomous reprogramming and spare a lot of time especially when we need to test quickly applications using many sensors. Unfortunately, this approach was not really functional, so we didn't use it in this project.

High Frequency Sampling This module, implements a non-standard timer with μs precision, but it is not precise enough to keep synchronized for long times of functioning. As we had already many data loss problem when sampling at higher frequencies than 500 Hz, we did not try this approach due to its experimental status. It is based on a new model of timer which has a μs accuracy instead of ms, but it becomes very quickly unsynchronized so we cannot sample more than some milliseconds.

Random This module provides us a simple random number generator and can be integrated easily into any application with a single line of code.

Multihop Networking Sensor networks are pervasive by nature : the quantity of nodes is almost boundless. Therefore a key to realizing this potential is multi-hop mesh networking, which enables scalability and reliability. A mesh network is a generic name for a class of regularly distributed networks of embedded systems that share several characteristics including :

- **Multi Hop** - the capability of sending messages peer-to-peer to a base-station by having nodes acting like routers, thus forwarding packets towards the base-station
- **Self-Configuring** - ability of network formation without any human intervention
- **Self-Healing** - on-line modification of the network structure
- **Dynamic routing** - ability to adaptively the route based on dynamic network conditions, e.g. link quality, hop-count and other metrics

These types of networks are perfectly suited for large-scale networks distributed over a geographical area. Since there are generally multiple routing paths between nodes, this model is robust to individual node failures. Certain nodes could be designated as *masters* or more commonly *group leaders* which are endowed with additional functionalities as for example performing arbitration in decision-making processes for the set of motes surrounding them. Another interesting example in our case would be to collect data from several neighbors and aggregate all the relevant data into a single packet and send it back to the base station. An advantage in homogeneous devices nets is that as all nodes are identical and have the same computing and transmission capabilities, so as soon as a group leader fails, another node can immediately take over these duties.

We might want to find an effective granularity for time period between two sensor readings. Large period (above 5 seconds) might almost unavoidably lead to miss short activity patterns not present at monitoring time, as a quick sound. In the other method, short monitoring period (less than 1 second), has the major disadvantage of consuming very much energy when there's nothing to monitor. If the motes are setup in a building where a network infrastructure is or may be available to connect the motes to (with the EPB when using Berkeley motes) so that energy is supplied through Ethernet connection and energy efficiency is not a constraint any more, the second approach is the one to use. But this physical infrastructure might be a much worse constraint. First, the price to pay will definitely be higher than simple wireless motes because you need an additional EPB per mote, and you need a wiring of all motes through Ethernet cabling which is very expensive too. The physical networking of the motes is a real problem for the system scalability, because the network infrastructure must be present if we plan to integrate additional motes to the system later, which can be done on the fly when using wireless communication. Another constraint concerning scalability is at the system startup where the initial routing tables are built according to the relative positioning of the motes, because the time needed for system booting is of complexity $\mathcal{O}(n^2)$, with n being the total nodes in the system. Depending on the sparsity of the nodes and the connectivity of the network, the routing problem in case of fully connected networks suffer from its NP-complexity, which makes it computationally intractable for large networks, even with huge processing power.

3.5 TOSSIM, a TinyOS simulator

TOSSIM is a powerful discrete-event wireless network simulator, capable of simulating a lot of different aspects of the real world environment, as for example transmission errors, energy level and many other things. But, in the framework of our project, TOSSIM is unable to simulate sounds and microphone reading and even less modelize sound propagation with reverbs, attenuations. In the practical case of analysis of the localization it's just useless. That is the reason why we didn't use it in our experiments. Instead, it becomes very useful for simulation of the algorithm itself, how it behaves in function of node density, communication interference, timing delays, node failures, and all aspects concerning the physical behavior of an algorithm. But as this is not the aspect we want to focus on as we try to stay generic and not application-specific, on and for that reason we won't talk about simulations any more.

3.6 Power Management

As we already suggested, the most precious resource is energy. As these nodes are intended to run autonomously and in some cases physical access to the device is impossible or would cost too much, it is important that each unit manages its energy efficiently in order to reach a maximal system lifetime. This aspect is probably against which design choices must be evaluated. There's a strong focus in current SN research on increasing lifetime through power generation, conservation and management. That led already to design of small *Micro Electro-Mechanical Systems* (MEMS) RF components for transceivers, including capacitors, inductors, etc. Another trust is in designing MEMS power generators converting the energy contained in solar radiations, vibrations (either electromagnetic or electrostatic), environment temper-

| Component | Active (mA) | Idle(mA) | Inactive (μA) |
|--------------|------------------|----------|----------------------|
| CPU core | 5 | 2 | 1 |
| Co-processor | 2.5 | 0.5 | 1 |
| RFM | 17.6 TX, 19.5 RX | - | - |
| EEPROM | 3 | - | 1 |
| LED | 4.6 | - | - |
| Temp | 1 | 0.6 | 5 |
| Photocell | 0.3 | - | - |

Figure 3.1: Current draw for each device on the MICAZ mote

ature, etc. Current batteries can provide roughly 1 J/mm^3 , while solar cells provide approximately $100 \mu\text{W/mm}^2$ in full sunlight, more than 100 nW/mm^2 indoors.

shows the current drawn by each hardware component under three scenarios: peak load when active, load in idle mode, and inactive. When active, the power consumption of the LED and radio reception are about equal to the processor. The processor, radio, and sensors running at peak load consume 19.5 mA at 3 volts, or about 60 mW, 100 mW if all LEDs are on. The mote itself consumes 8 mA when active and less than $15 \mu\text{A}$ in sleep mode. When in sleep mode the consumption is $20 \mu\text{A}$ with the voltage regulator on, and $2 \mu\text{A}$ without it. More noteworthy are the three sleep modes: idle, which just shuts off the processor, power down, which shuts off everything but the watchdog and asynchronous interrupt logic necessary for wake up, and power save, which is similar to the power down mode, but leaves an asynchronous timer running.

The minimum pulse width is $52 \mu\text{s}$, so this device consumes around $1.9 \mu\text{J}$ only to transmit a single bit. TinyOS manages power through the interaction of three elements. First, each service can be stopped through a call to its `StdControl.stop()` command. Second the **HPLPowerManagement** component sets the CPU into the lowest-power mode compatible with the current hardware state, given by the state of the I/O pins and control registers. This component interacts with the hardware directly at low-level, and sets the CPU power directly via the `adjustPower()` command. Third, the timer service can function with the processor mostly in the power-save mode. The basic idea is that, at the end of each data collection round, the mote calls the `StdControl.stop()` command in order to stop all the hardware that will be unused during the sleep time. Assuming that the mote must carefully listen to its neighbors and wake up as soon as a mote discovers something to listen to. We decided not to use a specific Active Message type to wake the neighboring motes, but the normal `Oscope` message type, in order to avoid extra energy expenses for dedicated messages. The drawback is that each message send by all local motes will trigger an event into each mote (even if it doesn't anything if the mote is not in sleep state), it's just extra spent energy because we must keep the radio on for all the time, and for that reason we decided finally to remove this function.

As the radio communication is by far the most expensive resource, a particular attention must be ported to the network protocol design. Many work in energy conservation methods had been discussed in [GW] and [CHE02], they use two criteria to optimize the energy consumption : the total energy consumption (TEC) of the system in order to complete a given task and the system lifetime (SL) that is the volume of the task that can be accomplished with a given energy level in the system.

There are many optimization major guidelines that can help in reducing the total consumption, as for example :

- Turn off unused transceivers at each time.
- Scheduling of node transmission to reduce packet loss and also avoid data redundancy in sent messages.
- Reduce communication overhead, i.e. defer communication when transmission channel conditions are poor.
- Use power control to transmit with a minimal energy sufficient to reach only the nearest neighbor. This also might reduce interference with the other on-going transmissions.

We notice that active radio listening also consumes a large amount of energy, while we don't need this feature to be active all the time. For that reason we might think that we must turn off the radio immediately after it has been used and keep it sleeping until the next utilization, which is not always possible. If the mote must be able to respond to asynchronous messages coming either from neighboring motes or from a base-station, we cannot simply turn off sensors and pray for a message to come only when the radio is active. In this case, we must ensure that all messages are allowed to be sent only at precise time steps. For doing that we need a robust and efficient synchronization algorithm with a μs granularity, which does not exist yet. When multiple nodes desire to transmit simultaneously, protocols are needed to avoid packet collision and thus losing data and energy. The first protocol designed for this purpose was proposed in the 70's at the University of Hawaii and was called ALOHA. Each node transmits a message when it desires, and if it receives an acknowledgement response everything is well. Else, the node waits a random time and retransmits its message. We can notice two major disadvantages, first the need of an acknowledge packet for confirmation which will double the network traffic and also increase the probability of congestion at each node in the network in the same time energy consumption doubles as well, and secondly as each node has to wait the returning packet, the other messages have to be buffered somewhere until the acknowledgement comes, which is a very bad idea regarding SNs storage limitations. We would also need to buffer all packets coming from the other nodes in the network if we use the multi-hop scheme. For that reason other protocols have been developed as the *Frequency Division Multiple Access* (FDMA) where different carrier frequencies are used *Code Division Multiple Access* (CDMA) and *Time Division Multiple Access* (TDMA). This latter scheme is the one of high interest in SN. The RF link is divided on a time axis and each node is given a predetermined time slot it can use for communication. This might decrease the sweep rate, but has the advantage to be easily implementable in software at the cost of fine and accurate time synchronization. The benefit yielded by this approach is to allow the mote to power down or "sleep" between its time slots and safely turn off RF module, thus radically decreasing its power consumption.

In our approach, we let the radio active all the time as we sporadically send time synchronization messages from the base station, we didn't investigate more deeply this aspect as it is not the purpose of this project.

The required transmission power increases as the square of the distance between source and destination, so multiple short-range message transmission might require less power than a single long-range one. A current research topic is *Active Power*

Control where each node collaborates with its neighbors in order to adapt its individual power level. This is what we call usually a *Decentralized Feedback Control Problem*. The power is changed through the command `SetRFPower()` provided by the **CC2420Control** component. The power radio power consumption can be dropped from 17.5 mA when at 0 dBm to 8.5 mA at -25 dBm, while the reception requires 19.7 mA.

3.7 Time Synchronization

In many network-centric applications, time synchronization is critical. Processing and propagation through different media with different physical properties (sound, RF, optical, etc.) takes time, collisions may occur, distributedness and asynchronous nodes.

For example, sensor fusion applications that combine a set of coincident sensor readings from different locations, such as in our case, require the establishment of temporal consistency of data. Several groups including UCLA, Vanderbilt and UC Berkeley have implemented time synchronization. The **GenericComm**³ component provides a command allowing to modify the data transmitted just as it transmits the first bit as the message is transmitted, that allows a sub-millisecond accuracy. Initial approaches to develop a low-level time synchronization component, have been unsuccessful. Many subtle and bad interactions observed with higher-level applications such as missed timer events. The real problem is that the time intervals which should be all equal (defined by the timer events) undergo some slight changes, becoming longer or shorter, thus weakening the real-time aspect, by preventing some critical event to reach completion before their deadline. Another approach, the current approach taken by TinyOS is to provide the mechanism to set the current system time (and timestamp messages at low-level), but the time when to invoke synchronization still remains to be defined by the application.

3.7.1 In-Network Processing

In-Network Data Processing refers to performing distribute the processing by sharing the computation with other nodes within the network as opposed to a central base station/user node. This allows to minimize the data to retransmit to the base station. The advantages of in-network processing are to reduce overall energy consumption as computation is cheaper than communication. The data aggregation theoretical approach found in [HSI⁺01] is interesting and it shows that data aggregation is able to reduce the network traffic up to 42% in some cases and reduce the total power consumption by a factor 3 to 5. We tried to implement such a dynamic node selection algorithm, but it turned out to fail under the high packet loss rate experienced in the physical deployment of our method, introducing an unstable system state such as deadlocks. It consisted of a 3 step algorithm. The first is the dynamic master selection upon acoustic event detection. Each mote detecting the event send a specific message containing its maximal *activation value* (maximal peak-to-peak distance), then each mote keeps track of the 3 most active sensors (without itself). After a timeout, generally 100 ms, the mote with the highest activation value will be elected as The Master, and it will broadcast a message containing the data sent too him by the three Chosen motes while the non-selected motes will sleep until the next synchronization

³This is basic component that is used for simple broadcast messaging.

messages arrives from the base station. Then the process is repeated until no more sound activity is detected. The same paper also shows that all packets need to be sent to the Master, who will aggregate all the data into single packets to send to the base-station. The problem is that the Master must have enough storage in order to temporally correlate the received data, and send in the same packet all the data that was read at the same time by all involved sensors. It is important for such a Master node to be able to deal efficiently with real-time constraints and have an efficient data redundancy removal mechanism.

These requirements lead to an interesting alternative, which is to use so-called *macro-nodes* regularly distributed over the whole network. They consist of motes attached to EP-RB and are thus physically wired through Ethernet to a base-station. This allows *Power Over Ethernet (POE)*, which removes the energy limitation and provides a more reliable communication channel endowed with a high bandwidth. The system designer needs to determine the relative ratio of macro-nodes over all nodes in the network. Increasing the number of macro-nodes in the network would definitely improve the application performance but simultaneously, the overall system cost would also increase.

Finally, the system needs to be made self-configurable. This means that the micro-nodes should organize themselves automatically into clusters based upon their proximity to a macro-node. And they should be able to offload computation onto the macro-node acting as the cluster head. Unfortunately, this approach is not always possible, as macro-node failures might be critical and could totally break the system.

The overall latency can also be reduced by having motes compressing data before sending it and hence reducing the transfer overhead. We work done in [MPS⁺02] suggests, common compression algorithms as Huffmann, Lempel-Ziv (gzip) or Burrow-Wheeler (bzip) might reduce the size of the packets to send from 20% to 60%.

3.8 Sound Detection

The basic and binary sound detection (sound or no sound) without taking care of its nature or the intelligent sound detection taking into account many physical properties of the sound, as frequency distribution and duration and nature of this sound, i.e. music, sinusoidal signal, human voice, noise removal, multisource separation, etc. are two very different tasks. As the complexity of the sound increases, the computing capabilities required to perform the analysis of such a sound increase as well. Due to the nature of this project, we wanted to stay as simple as possible so we dealt solely with the analysis of simple sounds, without regarding its properties, as energy distribution for example in order to determine the nature of the sound taken into consideration.

The qualitative aspects of a sound have been investigated a lot in different fields computer science, signal processing and voice recognition but using traditional devices : microphones and FFT of these signals. But, they have not yet been addressed within the Sensor Network frame because of its intrinsic limitations : low power devices with very limited computing capabilities, absence of real-time infrastructure and limited sampling frequency.

3.8.1 Frequency limitations in sensor networks

The accuracy of the location estimation is directly a function of the sampling frequency. The higher the frequency the finer the granularity and precision we can get. But the way sampling is performed on the motes clearly limits the higher frequency we can reach. As we explain in chapter 6, the best tradeoff we get The sampling on the mote is done as follows : we set a periodic timer to fire each $T_s = 1/f_s$ seconds, where f_s is the desired sampling frequency. When the timer fires, it'll signal an asynchronous event to the CPU and the task being served at that moment will be preempted in order to serve the event. At that moment, the event will send the command `ADC.getData()` to the *Analog-to-Digital Converter (ADC)* which will put the sensor reading data normalized onto 10 bits into a dedicated register. When the data is written (note that it takes some time), the ADC signals back an event to the CPU containing the data as parameter, and the program may use it directly. The problem is that the data is not read instantaneously, but some microseconds of delay might be introduced, and the higher is the frequency of the sampled sound, the more problematic are the delays.

Another problem using this approach is that the timer is set to a period given in milliseconds, thus the maximal sampling rate attainable with this method is 1 kHz. Depending on the application, that might be not enough, because of the Nyquist theorem which says that the sampling rate must be twice at least of the maximal desired frequency to detect. That means, with 1 kHz the maximal frequency we can accurately discern is 500 Hz. This is rather sufficient for basic sound detection, but not for precise sound spectrum analysis. If we want to pass the sampled signal through a FFT in order to see the wave energy for each frequency range, we will be limited to frequencies below 500 Hz. That prevents us from using sensor networks for real-time speech processing, as this requires a minimal sampling rate of 8 kHz (the telephone allows this frequency) in order to capture the major components of the speech signal. In professional speech processing applications, a frequency of 44.1 kHz or 48 kHz is required. Only with such high rates, we are able to capture the full spectra of sounds produced by humans.

In order to reach such frequencies with sensor networks, we might want to calculate the maximal sampling a mote allows. This is equal to the delay between the `ADC.getData()` procedure call and its `ADC.dataReady()` response event, and that will be the minimal frequency sampling rate that can be reached by making a loop which reads sensor data as soon as it is available. But in the practical framework of sensor networks this approach has two heavy disadvantages that are totally opposed to the usual philosophy of SNs. The first is that it will monopolize the CPU bandwidth, which prevents a correct functioning of the device by its incapacity of handling hard real-time constraints quickly enough, moreover that might imply a large variance of the time delay between two consecutive samples, so that won't be real sampling.

The second, which might be even more trivial, is that continuous polling of sensor and sending the data through radio communication is a very bad idea in terms of energy awareness. The radio activity, is one of the most power-consuming task in SN, sending one packet of data is equivalent to having 3000 instruction processed by the CPU in energy cost terms. Generally, the packets to send should not be too long because the larger is the packet, the higher is the cost if it gets lost. So as smaller packets are sent they also need to be sent quicker. Of course data must not accumulate into a buffer before being sent because delays will be introduced. If we wanted to sample at 8 kHz, we would need to be able to send 800 packets per second

continuously, it is clearly not applicable.

3.9 Implementation

Our goal was to find a method to quickly decide locally if what is heard by the microphone is relevant and if we really need to send it. After reviewing many methods of increasing complexity, we decided to tackle this question very simply because of the computing limitations on the mote. We simply make 10 readings at frequency f_s at different random times⁴ and if the peak-to-peak value of the 10 samples is higher than a fixed threshold, we infer that a relevant sound has been perceived. The algorithm was implemented as a simple finite-state machine with different functioning modes we explain under.

What is a relevant sound?

What is statistically a significant threshold in any case? Only the Oracle of Delphi can answer this question. But we, engineers, we can find a good criterion (heuristic) for a given and quite predictable environment. But false positive and true negative cannot be avoided in all cases. The most important for such a system is to avoid all true negatives, if possible (if we are in the museum case, it would be quite annoying if a Picasso is stolen with such high-tech security devices, because the noise made by the thieves was not considered as relevant). False positive is not really a problem if they occur very rarely, because it only involves a guard to go on place and see notice that it was a false alert. But if major military armada has to be deployed each time something is detected, that might be much more annoying. A first pace in this direction is to configure the microphone gain, which is done through the command `Mic.setgain()`. We need to find a threshold, should we turn the microphone into a very sensitive sensor, which is able to detect very small pressure fluctuations due to faint sounds and detect only when the sensor is really saturated. Or should we, on the contrary, make it less sensitive and conclude that there's sound activity at smallest fluctuation. We decide to turn a bit more sensitive in order to detect faint sounds.

3.9.1 Mote Algorithm

The detection task was implemented by a fairly simple code. The basic idea is that the mote sleep, and they wake up each 1 second for performing a monitor. By monitor, we mean make 10 successive reading of the microphone output, and if the maximal peak-to-peak value of these readings exceed a certain threshold, we conclude that there's significant activity in the area and then the mote samples. The controller of the mote has been implemented as a simple finite state machine, described in pseudo-code in Figure 3.3 and the corresponding FSM in Figure 3.2.

3.9.2 Central computer

In the approach we developed in this project, we use a base-station for interfacing purpose, i.e. a mote connected to a central computer connected which catches all packets sent in the network and may also send feed-back data to the motes in order

⁴We choose to make this random times prevents all motes to wake up at the same time and maybe not perceive short sounds which happened while the motes were sleeping.

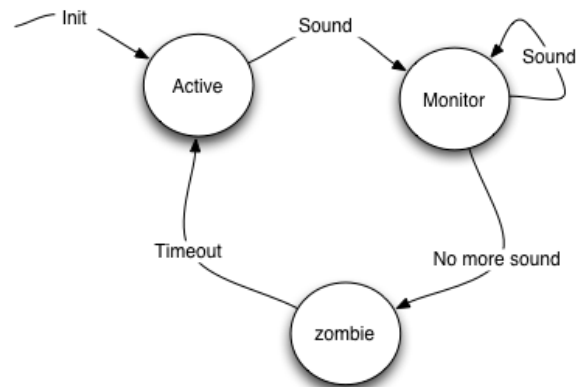


Figure 3.2: The finite state machine describing the controller of the program running on the mote.

```

initialization {
    input parameters {
        threshold;      /* Minimal Peak-2-Peak value for significance */
        f_s;            /* Sampling frequency */
        pi_range;       /* Interval where pi is randomly selected */
        /**/
    }
}

set monitoring timer MT in pi;

loop {
    if ((MT fires) or (message received from a neighbor)){
        start sampling timer ST;
        make 10 readings of the mic output at freq f_s;
        if (peak-2-peak > threshold) {      /* Something is detected */
            send data to base-station and time-stamp the packet;
            keep sampling;
        } else {
            stop the ST;
            set MT to fire in random pi;
        }
    } else if {
        reset timer.
    }
}

```

Figure 3.3: Algorithm running on the motes

to reconfigure them online, that enhances dynamism of the system by providing almost real-time adaptation. Of course, our approach does not fit with the Swarm Intelligent (SI) assumptions that states that the system has access only to local knowledge and perception, thus denying the existence of a global signal given by a central coordinator.

In our approach, the base-station has three important roles:

Data retrieval The central computer has the purpose to gather all the data coming from the network that is all sensor readings sent by the motes, statistics concerning the radio transmission and routing tables, internal states and timing of all motes, and this data must be stored in a convenient way in order to be processed and then closing the feedback loop by sending commands to the network (individually or broadcasted for all motes) in function of the internal state of the sensor network. This part is done by a modified version of the original Oscilloscope Java program included with TinyOS. Its purpose is to store all data coming from the network directly into a Matlab .m file for later processing. Our program has two functioning modes, a static mode which solely plots the sound wave structure in (almost) real-time upon each packet arrival. The second one is the dynamic mode, whose goal is not only to plot the wave as in the static mode, but also logging the data received in a Matlab script. We called this mode dynamic because all data received is not saved in a single huge file, but rather a fixed time window is used that creates independency of data between two contiguous time windows. Of course, a different Matlab script will be generated for each time window. These time window method yields two main advantages. First, as the selected motes are the first to send the data back to the base station, it allows to dynamically switch the sensors used for the localization according to the displacement of the sound source within the environment, at each time window. The second advantage is that, as we used the approach of a global signal for time resynchronization, we decided to send the message at the end of each time window.

Data processing Once we have the file, we have a very flexible representation of the sound signal and its properties, ready to be loaded into Matlab and be directly fed to the different sound processing functions. They are stored to be processed later by the tools we developed in Matlab. Our initial plan was to create a single program in Java since the newer versions of Matlab allow direct interpretation of Java code into Matlab and vice-versa. We planned to use this ability in order to get the packets received from the network and to process them in order to localize sound sources in real-time. But each time we tried to connect to the base station from Matlab, either it didn't work at all if we take packets at a low-level, either Matlab crashed directly when I tried to use a high-level Java application as an intermediate by reading packets which are processed and fed them to Matlab. We suspect the origin of this problem being the Java tools that come with TinyOS, to be adapted for the version 6.5 of Matlab, while we are using version 7. So we weren't able to make a single program and that's the reason why we made two different programs. We plan to further continue the development of the application by making a much more complete version of the software, which is able to retrieve network data and perform localization in real-time into a single program.

Interaction with the network This latter part, less interesting for the theoretical part, but very important in a practical implementation is the ability to interact with

the nodes and send them individually or broadcast specific orders in order to re-configure dynamically every tuning of the network during worktime, as for example changing the RF power, changing their sampling frequency, etc.

Chapter 4

Sound Perception

4.1 Human Auditory System

Understanding how humans hear is a complex subject involving the fields of physiology, psychology and acoustics, nonetheless we'll try to present a short overview on how the human ear serves as a transducer, converting sound mechanical energy to nerve impulses which are transmitted to the brain thus allowing us to perceive the pitch of sounds by detection of the wave's frequencies, the loudness of sound by detection of the wave's amplitude and the timbre of the sound by the detection of the various frequencies which make up a complex sound wave.

The ear consists of three basic parts - the outer ear, the middle ear, and the inner ear, each of them serving a specific purpose in the task of detecting and interpreting sound. The outer ear serves to collect and channel sound to the middle ear. This intermediate part transforms the sound wave energy into internal vibrations of the bone structure of the middle ear and finally transform these vibrations into a compressional wave in the inner ear. The inner ear serves to transform the energy of a compressional wave within the inner ear fluid into nerve impulses which can be transmitted to the brain.

The outer ear consists of the ear flap (usually called *pinnae* or *auricula*), which is a cartilaginous structure covered with skin. Pinna role is capital for sound localization into vertical plane by echoing sound waves on its faces thus directing sound waves to the *auditory meatus* differently according to the sound source location. The meatus is an approximately 2 cm long ear canal which will channel the sound waves to the eardrum, a tightly stretched membrane at the interface between external and middle ear. As sound travels along the auditory canal, it is still in the form of a pressure wave which hits the eardrum. The eardrum will move according to the energy contained into the sound, which is converted into vibrations of the inner bones structure of the ear. The inner ear is an air-filled cavity containing three tiny, interconnected bones - the hammer, anvil, and stirrup. A compression forces the eardrum inward and a rarefaction forces the eardrum outward, thus vibrating the eardrum at the same frequency as the sound wave. Being connected to the hammer, the movements of the eardrum will set the hammer, anvil, and stirrup into motion at the same frequency of the sound wave. The stirrup is connected to the inner ear; and thus the vibrations of the stirrup are transmitted to the fluid in the of the middle ear through a compression wave within the fluid. It is separated from the inner ear by a bony partition which



Figure 4.1: Human auditory system.

contains two windows the oval window (*fenestra vestibuli*) and the round window (*fenestra cochlea*). The three tiny bones of the middle ear act as levers to amplify the vibrations of the sound wave. Furthermore, since the pressure wave striking the large area of the eardrum is concentrated into the smaller area of the stirrup, the force of the vibrating stirrup is nearly 15 times larger than that of the eardrum, thus enhancing our ability to hear the faintest of sounds.

The inner ear consists of a cochlea, the semicircular canals and the auditory nerves. Semicircular canals serve only as an accelerometer for maintaining equilibrium, but they are not involved in sound perception processes so we will not mention them anymore. The cochlea and the semicircular canals are filled with a water-like fluid. The cochlea is an approximately 3 cm long snail-shaped organ. The inner surface of the cochlea is lined with over 20 000 hair-like nerve cells which perform one of the most critical roles in our ability to hear. These nerve cells differ in length by minuscule amounts; they also have different degrees of resiliency to the fluid which passes over them. As a compressional wave moves from the interface between the hammer of the middle ear and the oval window of the inner ear through the cochlea, the small hair-like nerve cells will be set in motion. Each hair cell has a natural sensitivity to a particular frequency of vibration. When the frequency of the compressional wave matches the natural frequency of the nerve cell, that nerve cell will resonate with a larger amplitude of vibration. This increased vibrational amplitude induces the cell to release an electrical impulse which passes along the auditory nerve towards the brain. In a process which is not clearly understood, the brain is capable of interpreting the qualities of the sound upon reception of these electric nerve impulses.

Response of the basilar membrane to sound

The basilar membrane has two structural properties that determine its response to sound, the wideness and the rigidity. The Hungarian-American biophysicist Georg

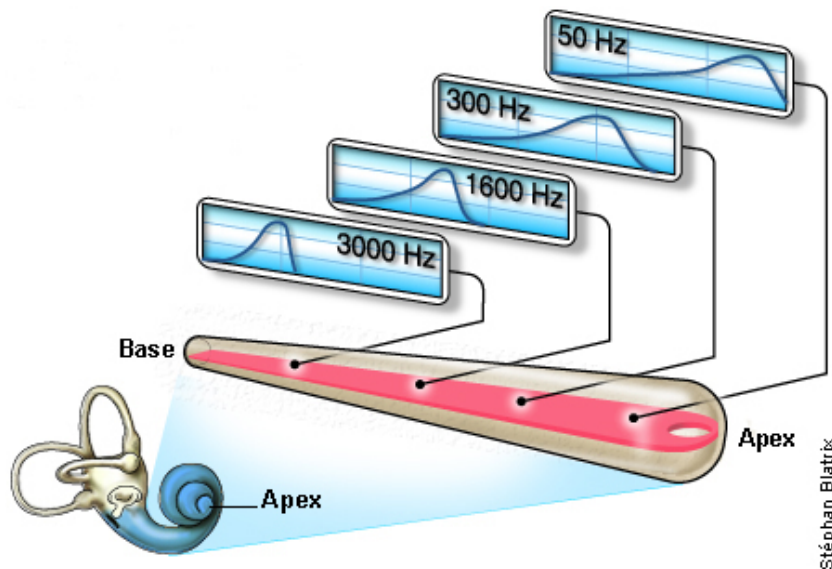


Figure 4.2: Tonotopicity of the basilar membrane.

von Békésy determined that as sound causes a continual push-pull motion of the footplate, the movement of the endolymph makes the basilar membrane bend near its base, and that will create a wave that propagates toward the apex, like a wave on a rope when you give a snap while holding its extremities. The distance this wave propagates along the membrane depends on the wave frequency. The higher the frequency the higher the amount of the initial energy dissipated for moving the stiff base of the membrane, in which case the wave won't propagate very far. On the contrary, low-frequency sounds will generate waves that will propagate much further before dissipating all their energy. These locations of different membrane responses establishes a coding which is responsible for the neural coding of pitch, as we'll see later. To this point, we only discussed about the mechanical wave transformations that occur in the middle and inner ear. Now, we need to see how are these membrane deformations transformed into neural signals. The auditory receptors cells are named the *organ of Corti*¹ and are principally composed of hair cells, rods of Corti and various supporting organs. The critical event that transforms sound into a neural signal is the bending of these hair cells induced by the basilar membrane movement, which determines the opening of the K^+ and Na^+ ion channels.

4.2 Human and Animal Sound Localization

At the beginning of the century, Lord Rayleigh showed that for low frequency sounds, directional information is contained in the difference in the phases of the sound waves at the two ears, the so-called *Interaural Phase Difference (IPD)*. Later in 1948, Jeffress proposed that the IPD is generated by an array of coincidence detector neurons

¹named after the Italian anatomist who first discovered them

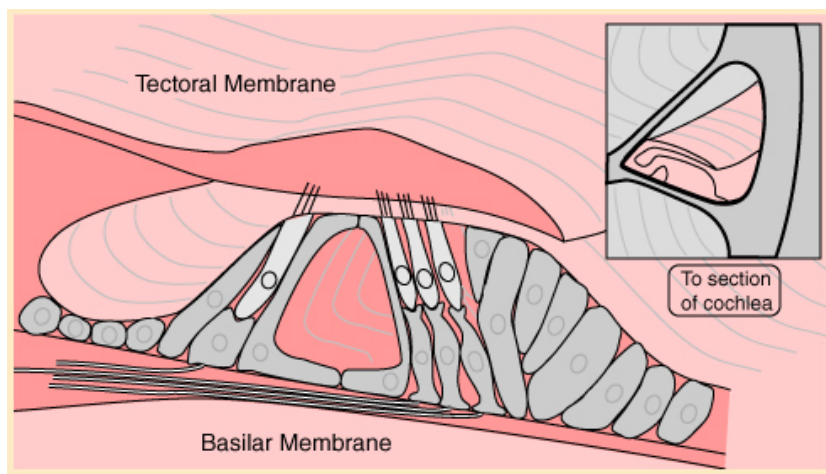


Figure 4.3: Hair cells of the basilar membrane.

tuned for different ITDs. This tuning is defined by the difference in axonal conduction delay from each ear. Each of these neurons receives input from both ears through axons which act as delay lines, with the difference in the delays associated with the left and right connecting axons, varying systematically across the array of neurons. Sound from a particular direction will reach the closer ear first, so there will be an IPD which is a function of the direction of the sound. This is called *delay line* hypothesis. The signals from the two ears will arrive at different times, and give rise to a firing rate which is zero or very small. For neurons whose internal delay corresponds to the IPD between both ears, both signals will arrive at the same time thus producing a high firing rate. The location of these neurons within the array identifies the direction of the sound. Almost all experimental and computational studies on sound localization are built upon the basic Jeffress picture. Many experiments carried out at Caltech by Konishi and Carr [Kon73, KC90] show experimental evidence for such a delay lines system and the associated topographic layout of neurons in certain birds. Computational studies done in by Gerstner et al. [GKvHW96] show how a neural network might "learn" the proper connections to perform localization with very high accuracy. However, recent study in Mongolian gerbils, do not confirm the existence of such delay lines in mammals [MG03].

The majority of neurons composing our auditory system are sensitive to stimulus frequency. A yet unanswered question is : How is frequency represented in the nervous system?

Tonotopy This ability is directly related to mechanical properties of basilar membrane, whose deformation depends on the properties of a sound wave. A similar representation is done in auditory nerves, the hairy cells near the membrane basis c.f. Figure 4.3 are sensitive to specific high-frequencies, while hairy cells near the apex are sensitive to low-frequency sounds. That makes a systematic mapping from cells localization in the cochlear nucleus and their characteristic frequency. In other words, there is a map of the basilar membrane at the cochlear ganglions level. This

systematic organization of an auditory system structure based on frequency is what we call *tonotopy*. That is similar to retinotopy in the visual system. Due to this property, localization of an activated neuron in the auditory system indicates sound frequency. Nevertheless, for two reasons, it is necessary for the frequency to be encoded else than just by maximal activation areas on the tonotopic map. The first reason is that there aren't any neurons with a characteristic frequency under 200 Hz. The other reason is that basilar membrane deformation depends on the intensity and not only on frequency. A more intense sound will bend the membrane at another place.

Phase correlation The other major information resources, thus complementing information given by tonotopic maps, is the relation between sound duration and neuronal firing. Auditory nerve neurons analysis shows that there's a strong correlation between the firing pattern and the phase of the sound wave. That means that for low-frequency sounds, there are neurons who generate action spikes at the same time sound wave reaches a specific phase. In this case frequency is similar to neuronal spiking rate. It may also be possible that the spikes are not generated at each cycle, but are still correlated with sound wave. With some neurons of this type, at least one of them firing each cycle we can easily measure sound frequency. This type of correlation happens with sound waves with frequencies up to 4 kHz. Above this frequency, the spikes are randomly generated and are not correlated anymore with sound wave phases. That is due to internal specification of neurons, and that they are limited to a maximal rate due to inhibitory periods. Frequencies above 4 kHz are represented only by tonotopy.

4.3 Sound localization in biology

The most evident clue for sound localization is the time needed by a sound to reach each ear. As the average distance separating ears is approximately 20 cm, sound may take 0.6 ms to travel from one ear to another. That means there's a simple relation between sound localization and interaural delay. This delay is detected by specific neurons in the brain stem thus allowing to localize the sound source. When we need to localize a source continuously emitting sound, things are getting worse because we cannot rely on onset difference time between the 2 ears. A reliable cue we can get in this case is to measure the phase difference between the signals presented at each ear. Let's assume a 200 Hz sound whose wavelength is approx. 172 cm, which is larger than distance between ears, so this cue is a reliable one. Now if a 20 kHz sound is presented, when a peak reaches left ear, it will need much less than 0.6 to be detected by other ear. In fact there are many peaks appearing between the two ears. In this case there's not anymore the straightforward mapping between the sound localization and the *Interaural Time Difference (ITD)*. Brain also uses a different process for localization : *Interaural Level Difference (ILD)*. However, this cue depends on the head shape and size as it acts as a barrier for sound propagation.

In summary, there are two main components in the sound localization process in the horizontal plane : from 20 to 2000 Hz ITD is mainly used and for frequencies between 2 and 20 kHz, the ILD are preferred. The combination of these processes is what we call the *Duplex Theory*.

4.3.1 Binaural neurons sensitivity

It has been shown that cochlear nuclei receive the projections of auditory nerves located on the same side, which means that these cells are all monaural. But, the next steps of auditory information processing are associated with binaural neurons. Their response characteristics suggest that they have an important role into sound localization process. The superior olivary is the main repository of such binaural neurons. Although relations between the activity of binaural neurons and sound localization is still controversial, the correlations are indisputable. The olivary neurons receive information from the cochlear nuclei of both sides of the body, by neurons which respond with a phase correlation for low-frequency sounds. Olivary neurons immediately evaluate the interaural delay between both ears. How is neuronal circuitry able to make neurons responds to interaural delays? Nature found a solution to this problem by using the axons as delay lines allowing to accurately measure these small delays. A sound reaching left ear, generates a chain of action potentials in the corresponding nuclear cochleii, which are forwarded to the olivary. After some time, the sound wave will reach the other ear thus generating another chain of action potentials in the other cochlear nucleus. Due to the specific organization of the axons in the olivary, these action potentials will reach their target after different delays. When both potentials meet at a specific neuron (the ITD is compensated by a longer path to travel for the ipsilateral potentials), they generate *Excitatory Postsynaptic Potentials (EPP)* whose sum is more important than a single EPP generated by the activation of a single ear. There is plenty of cells programmed to respond to specific ITD. In order to measure such fine time delays, the neurons composing the auditory system are tuned to these high-speed operations, and generate action potentials faster than others neurons do. The second mechanism is also implemented, which makes our neurons sensitive to ILDs. Neurons of type EE are moderately excited by a single ear, but their activity is much higher when both ears are activated simultaneously. The neurons of type EI, are excited when sound reaches one ear and are inhibited when they reach the other ear. This EI neurons are very interesting, because they are particularly sensitive to ILD, because the excitatory effect of one ear is combined with the inhibitory effect of the other. That is very useful for sound localization at high frequencies.

4.3.2 3D Localization

Localization of a sound source in a three-dimensional plane, has already been intensively studied a lot, but in this project we deal exclusively with localization into the horizontal plane, without computing the elevation of the sound source. As azimuth is easily derived from ITDs and ILDs, elevation cannot use these cues too, because they are independent of the elevation. However, humans are able to localize the elevation of a sound source by using other (monaural) cues given by physical structures as for example the pinnae, torso and shoulders, which reflects the sound differently according to source elevation. Owls, which have their ears in front of the head and not on the side as humans usually do, rely on the cues provided by the remarkable vertical asymmetry of the placement of their ears, which makes ILD vary also with the elevation of the sound source. Moreover, they also have sound-reflective feathers which play a similar role to the pinnae for humans, reflecting sounds differently according to their elevation. It's interesting to note that the cues provided by the pinnae are only useful for sound above 3.5 kHz, while torso introduces important cues between 0.7 and 3.5 kHz.

The reason we explained how does human and animal auditory system work, is to show how efficient are the processes involved into animal sound localization and how are the neuronal structures used as substrate. It could be very useful to have such handwired systems as for example a VLSI chip dedicated to sound localization, compared to a software emulating the localization processes. The work done by Pu [Pu98] is directed towards such a neuromorphic microphone. This hard-wiring of the whole localization part, suggest us that it would be interesting to design a special sound localization sensor board that can be plugged to a MICAZ mote and that will handle the auditory signal so letting TinyOS not care about sound processing thus increasing its responsivity to real-time.

Chapter 5

Localization and detection

5.1 Sound localization

Sound localization refers to the law (or rule) between the spatial location of an auditory event and characteristic cues concerning that event. This process heavily relies on the analysis and interpretation of the information contained by specific source attributes as its temporal and spectral activity. There exist two main categories of cues which are related to the sound source itself in the first case and to the listener in the second one. The first category has two principal types of cues :

Spectral This cue is the spectrum of the emitted sound, that is the representation of a signal in time and frequency domains, which is called a *spectrogram* and can be obtained using a Fourier Transform as will be explained in the following sections. Localization results can be strongly influenced by the spectrum of a sound and the results of the process may vary whether an emitted sound is a pure sine wave with a narrow spectral width centered around a *Central Frequency (CF)*, or a complex sound with a broad spectral width. We can also use the distribution of spectral energy, the "shape" of the spectrum in order to draw relevant conclusions upon the source location, as the emitted energy distribution in time and frequency conveys information upon source characteristics. A "loud" source might be localized at a different position than a quiet one.

Temporal This type refers to the "temporal shape" of the sound. Some sounds have what we usually call *onset¹ and offset² transients*. That means there is a lot of dynamic movement taking place at the extremities of the sound signal. Cymbals are an example of signal containing such onset transients, opposed to sound with more a static level, e.g. a sine wave or the sound of a flute or an organ. Both classes might also be classified differently according to the localization method. Finally, the duration of the sound signal can also influence the localization.

The second big category of cues are related to the sensory information that a physical agent perceives. The perception differs generally according to the agent shape and sensory system (type and quantity of sensors and their placement on the body).

¹Refers to the beginning of a sound event, i.e. when pressure starts to change according to a sound wave.

²refers to the end of a sound event

Thus the signals received at each sensor can vary differently upon the spatial location of the source (azimuth and elevation) and depending on the way it is reflected by the environment and the obstacles the sound waves encounter.

Interaural Cues These refer to the relation of the signals perceived at the same time by two sensors located at different locations. These are the main types of cues we'll use throughout this project. They consist of the *Time Difference of Arrival (TDOA)* of a sound wave at the two sensors or the difference in amplitude between the perceived signals.

HRTF Cues As sound waves interact with objects they are scattered differently according to their physical properties. For example in the human auditory system, the sound waves are transformed by the head, the body and more particularly the external ear, before they reach the eardrum, and thus they are propagated differently according to the surfaces they encounter and the way these reflect the sound wave. In the opposite of interaural cues as time difference, these cues are different for each person and they depend solely on the morphological properties of the subject. Nonetheless, they can be experimentally measured by setting microphones into each ear and measure the two signals perceived for different azimuths and elevation. We can model such a binaural signal filtering using what is usually called a *Head Related Transfer Function (HTRF)*.

Dynamic Cues These cues arise from the movement of the head or the whole body, in order to change the orientation of the head relative to the sound source. This shift in perceived pattern modifications, can be used in order to get more information concerning the location.

Vision Cues At last, but not the least important cue (although not used by all animals) is the visual cue, and it corresponds to the visually perceived apparent position. Owls do rely a lot on these cues, and they create a tonotopic mapping between acoustic and vision during the two early months of their developmental stage. This process was suggested by the work done by Knudsen and later by Konishi, where they showed that owls with an ear blocked since their birth systematically shift the mapping between source location using auditory system and the location of this source within their visual field.

5.1.1 Localization techniques

Many solutions have been proposed to tackle the problem of a passive (non-moving) sound source. Most of them usually rely on the acquisition of time-delayed replicas of a source signal at spatially distributed sensors and follow a common *two-step method*. The first step usually takes signals received at each sensor and computes a cross-correlation on it in order to find an explicit approximation of the time delay. This value is then used to geometrically determine the *Angle of Arrival (AOA)* of the sound at the midpoint between the sensors using simple geometrical calculus. The second step might also be more complex than a simple geometrical function of the time delay. For example a neural network might be used for the classification of the time-delay into the class corresponding to the most appropriate angle. Other models do not use the explicit representation of the time delay estimation and are thus able

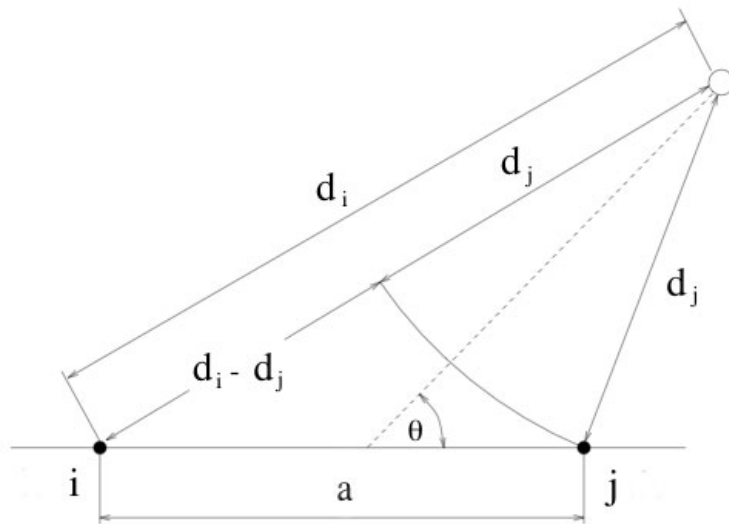


Figure 5.1: Geometric representation of the system composed by binaural sensors and the sound source.

to reduce errors significantly. Finally another class of methods are directly inspired by the human auditory system whose basilar membrane reacts to the different frequency components and the corresponding neuron groups for each location encodes the signal in their respective frequency bands. These models, also called *Auditory Filter Models*, decompose the signals perceived at two sensors into spectral components corresponding to different overlapping frequency bands. This is done by a simple multiple filter bank. The resulting representation in time and frequency is called a *neural spectrogram* or more commonly *cochleagram*. With this method we are able to get a non-uniform frequency resolution, and might contain more information as with the other methods and thus increased localization accuracy. But the frequency resolution is rather poor compared to resolutions that can be attained with time-frequency analysis computational methods (e.g. STFT), since the number of filters (and thus perceptual bands) usually vary between 20 and 50 filters. These methods are interesting since they are able to analyze energy distributions among frequencies, and can be used to improve the sound localization. This filtering can also be used to segregate multiple sound sources by isolating different frequency components, as described in [Vis04]. Unfortunately, this work will not focus on this specific case general case, hence from now on we'll deal only with a single source present.

5.2 The Situation

Now we'll describe the geometric situation of a single sensor pair and give a formal definition of the concepts we'll use throughout this project.

On 5.1, the small black dots denoted i and j represent the two sensors and d_i , d_j are their respective distance from the sound source denoted by a grey circle. We denote the distance from the source to the midpoint between the sensors as ρ and so

we can say that $d_i = \rho - \Delta\rho$ and $d_j = \rho + \Delta\rho$ where $\Delta\rho = \frac{d_j - d_i}{2}$.

Sound localization task consists of estimating the angle θ with as much accuracy as our tools allow. Our convention states that a pair has always a left and a right element, and the angle $\theta = \pi$ if the source is on the left side of the left sensor i on the ij axis and $\theta = 0$ if the source on the right side of the right sensor j . The exact value of the delay D as given by a simple model for sound propagation is

$$D = \frac{|d_i - d_j|}{c}. \quad (5.1)$$

The method we use requires the assumption that the distance d from a source to any sensor is much larger than the distance between the two sensors a , that is $d \gg a$. In this case, both sensors will see the source within an azimuth angle approximately the same $\theta_1 = \theta_2 = \theta$. Another important assumption, especially when we deal with stationary signals, is to ensure that the distances between both sensors is less than half the wavelength of the emitted sound, in order to avoid phase ambiguities due to the fact that for a 2π -periodic signal $s(t) = s(t + 2\pi)$. Under this assumption, we can approximate the signal arriving at the two microphones by a plane wave and in this case we can define the *Interaural Time Difference (ITD)* as

$$D = \frac{a}{c} \cos(\beta) = D_{MAX} \cos(\beta) \quad (5.2)$$

where $c = 344$ m/s is the speed of sound in the air in normal conditions $D_{MAX} = \frac{a}{c}$ is the maximal delay possible between the two sensors. The method we use estimates the time delay and replaces it in equation 5.2 in order to derive the angle of arrival.

The sound intensity is inversely proportional to the square of the distance, so the *Interaural Level Difference (ILD)* (in dB) between the two sources equals

$$\Delta L = \log \frac{d_i}{d_j} = \log \frac{\rho - \Delta\rho}{\rho + \Delta\rho} \quad (5.3)$$

The ILD is much more complex to compute because it varies from person to person due to the shadowing effects of the head and ear and depends on frequency, azimuth and elevation. This can be modeled using the concept of HRTF we explained in the previous section. A very simple model is

$$ILD = \Delta L(\theta, f) \approx \alpha_f \cos \theta \quad (5.4)$$

5.3 Mathematical Model

A signal emanating from a remote source and monitored in the presence of noise by two different sensors can be mathematically modeled as

$$\begin{aligned} x_1(t) &= s(t) + n_1(t) \\ x_2(t) &= \alpha s(t + D) + n_2(t) \end{aligned} \quad (5.5)$$

where $x_1(t)$ and $x_2(t)$ are the perceived signals by sensor 1 and 2 at time t , $s(t)$ is the real signal as generated by the sound source, it is assumed to be uncorrelated with

noise $n_1(t)$ and $n_2(t)$. The D is the real delay given by 5.1 and α is the attenuation due to energy diffusion during propagation through air. If the source is far away relative to the distance between the microphones, we can assume that $\alpha \approx 1$.

5.4 Fourier Basics

In this section we'll describe briefly the basic theorem concerning Fourier methods used in signal processing. Given a periodic function $g(t) = g(t + nT)$, where T is the period and $f = 1/T$ is the fundamental frequency. The Fourier series of this function expresses it as an infinite sum of sine/cosine function with functions multiple of f :

$$g(t) = \sum_{k=-\infty}^{\infty} G[k]e^{i2\pi kft} \quad (5.6)$$

where the amplitude/phase is settled by

$$G[k] = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} g(t)e^{-i2\pi kft} dt \quad (5.7)$$

We can extend this notion to any function by take the limit $T \rightarrow \infty$ thus getting an infinite range of frequencies, we get the Fourier transform

$$G(f) = \int_{-\infty}^{\infty} g(t)e^{-i2\pi kft} dt \quad (5.8)$$

that is the representation of the $g(t)$ signal in the frequency domain, and we can recover the original function in the time domain by an *inverse transform*

$$g(t) = \int_{-\infty}^{\infty} G(f)e^{i2\pi kft} df \quad (5.9)$$

5.4.1 Discrete Time Fourier Transform (DTFT)

In practical cases due to digitalization of sound by sampling we know the function $g(t)$ only at some evenly space points $g(nT)$, where $T = 1/f_s$ and f_s being the sampling rate, the Fourier transform is given by

$$G(f) = \sum_{n=-\infty}^{\infty} g[n]e^{-i2\pi fnT} \quad (5.10)$$

note that this function has a period f_s . We can recover the sampled signal by the inverse transform

$$g[n] = \frac{1}{f_s} \int_{-\frac{f_s}{2}}^{\frac{f_s}{2}} G(f) e^{i2\pi f n T} df \quad (5.11)$$

5.4.2 Discrete Fourier Transform (DFT)

Now if we have a finite sequence $g = g[0]..g[N - 1]$ containing N samples from a function $g(t)$, the DFT is defined by

$$G[k] = \frac{1}{N} \sum_{n=0}^{N-1} g[n] e^{-i2\pi \frac{nk}{N}} \quad (5.12)$$

In this case the transform function becomes

$$g[n] = \sum_{k=0}^{N-1} G[k] e^{i2\pi \frac{nk}{N}} \quad (5.13)$$

which can be efficiently evaluated by the Fast Fourier Transform (FFT) algorithm.

5.4.3 Short-Time Fourier Transform (STFT)

This the analog version of the DFT defined as

$$X(k, f) = \sum_{t=0}^{N-1} w_a(t) x(t + kH) e^{-\frac{i2\pi t f}{L}} \quad (5.14)$$

where $w_a(t)$ is the analysis window of length L . In general we use a rectangular window. The complex values of the STFT are denoted spectral coefficients $X(k, f)$, and are indexed in time k and frequency f . The time is related to the hop-size H that describes how much a window moves from one hop to another. As the spectral coefficients are complex numbers, they can be represented as magnitude and phase, which is a very intuitive representation with respect to physical propagating waves.

In comparison to the spectral analysis performed by the human auditory system, the STFT enables different frequency resolutions. By choosing a longer window, it is possible to get a much higher resolution than that of the auditory filters, that means we are able to individually detect closely spaced spectral components that fall within the same perceptual band. Unfortunately, this increased frequency resolution comes at the expense of a decreased time resolution since the frequency estimates are averaged over a longer period and fine fluctuations might be lost. Moreover the length of the window is limited by the duration of the auditory event itself. When multiple sources are present, their energy emissions in time and frequency may overlap, and in that case no time-frequency analysis can isolate the two different sources, this is the fundamental principle of the time-frequency uncertainty. Moreover the methods we just discussed assume that the partials assume a constant frequency over the duration of the analysis window.

5.5 Localization Techniques

The most common method used to determine the time delay is to compute the cross-correlation function, which is defined as

$$R_{x_1x_2}(\tau) = E[x_1(t)x_2(t - \tau)] \quad (5.15)$$

where E is the mathematical expectation operator theoretically defined as

$$R_{x_1x_2}(\tau) = \int_{-\infty}^{\infty} x_1(t)x_2(t - \tau)dt \quad (5.16)$$

The estimate of the true delay is provided by the argument τ that maximizes 5.15. Since we observe the signals only during a finite time window, $R_{x_1x_2}(\tau)$ can only be estimated. This estimation is given by

$$\hat{R}_{x_1x_2}(\tau) = \frac{1}{T - \tau} \int_{\tau}^T x_1(t)x_2(t - \tau)dt \quad (5.17)$$

The factor in front of the integral is used to normalize the signals between -1 and +1 (assuming that the signal $x(t)$ is also in this range) in order to have a metric of the quality of the cross-correlation, but we did not use this factor in our implementation, as the amplitude can change significantly and the sampling is very low.

After replacing the signals 5.5 into 5.17, and considering that $n_1(t)$ and $n_2(t)$ are uncorrelated, equation 5.17 can be rewritten as

$$\hat{R}_{x_1x_2}(\tau) = \int_{\tau}^T s(t)s(t + D - \tau)dt \quad (5.18)$$

It is clear that this function will exhibit a maximum at $\tau = D$. Therefore, one way to estimate the time delay is by generating 5.16 numerically and calculate the time where the maximum is achieved. In practice, the signal is sampled at a certain frequency $f_s = \frac{1}{T_s}$ and an approximation of the cross correlation is given by a discrete time version

$$\hat{R}_{x_1x_2}(\tau) = \sum_{k=0}^N x_1(kT_s)x_2((k - i)T_s)dt \quad (5.19)$$

where N is the number of samples, that is $N \cdot T_s$ is the length of the time window under consideration.

The localization accuracy can be improved by pre-filtering the signals. Signal x_i may be filtered through H_i to yield y_i . The selection of the filters needs some *a priori* information. For example if the role of the filters is to accentuate the signals at those frequencies at which the SNR is lowest, then the filtering is expected to be a function of the signal and noise spectra that must be either known or estimated.

Let's denote the filter weighting

$$\psi_g(f) = H_1(f)H_2^*(f) \quad (5.20)$$

where * denotes the complex conjugate.

The cross correlation between $x_1(t)$ and $x_2(t)$ is related to the cross power spectral density function by the well-known Fourier transform relationship

$$R_{x_1x_2}(\tau) = \int_{-\infty}^{\infty} G_{x_1x_2}(f)e^{j2\pi f\tau} df \quad (5.21)$$

where the cross power spectral density is given by

$$G_{x_1x_2}(f) = \int_{-\infty}^{\infty} x_1(t)x_2(t)e^{j2\pi ft} dt \quad (5.22)$$

When signals have been filtered, the cross power spectrum between the filter output becomes

$$G_{y_1y_2}(f) = H_1(f)H_2^*(f)G_{x_1x_2}(f) \quad (5.23)$$

$$R_{y_1y_2}(\tau) = \int_{-\infty}^{\infty} \psi_g(f)G_{x_1x_2}(f)e^{j2\pi f\tau} df \quad (5.24)$$

If we prefer to use pulse instead of frequency in the Fourier transform (as used in modern literature on the subject), the final estimate of the delay is

$$\hat{\tau} = \max_{\beta} \int_{-\infty}^{\infty} W(\omega)X_1(\omega)X_2^*(\omega)e^{j\omega\beta} d\omega \quad (5.25)$$

We assume that $X_1(\omega)$ and $X_2(\omega)$ are the Fourier transform of $x_1(t)$ and $x_2(t)$. Note also that we passed from time to frequency domain through this transform, while $W(\omega)$ is equivalent to the filter weighting $\psi_g(f)$ defined above.

These method were proposed in 1976 by Knapp and Carter in [KC76]. The simplest filtering is called the *Generalized Cross Correlation (GCC)*, and consists of weighting the signals with this simple filtering :

$$W_{CC}(\omega) = 1 \quad (5.26)$$

Other different weighting functions $W(\omega)$ are generally used :

$$W_{ML}(\omega) = \frac{|X_1(\omega)| \cdot |X_2(\omega)|}{|N_1(\omega)|^2|X_2(\omega)|^2 + |N_2(\omega)|^2|X_1(\omega)|^2} \quad (5.27)$$

$$W_{PHAT}(\omega) = \frac{1}{|X_1(\omega)X_2^*(\omega)|} \quad (5.28)$$

We can notice that the ML approach require knowledge about the spectrum of the microphone dependent noises, while the PHAT doesn't, and hence has been employed more often due to its simplicity.

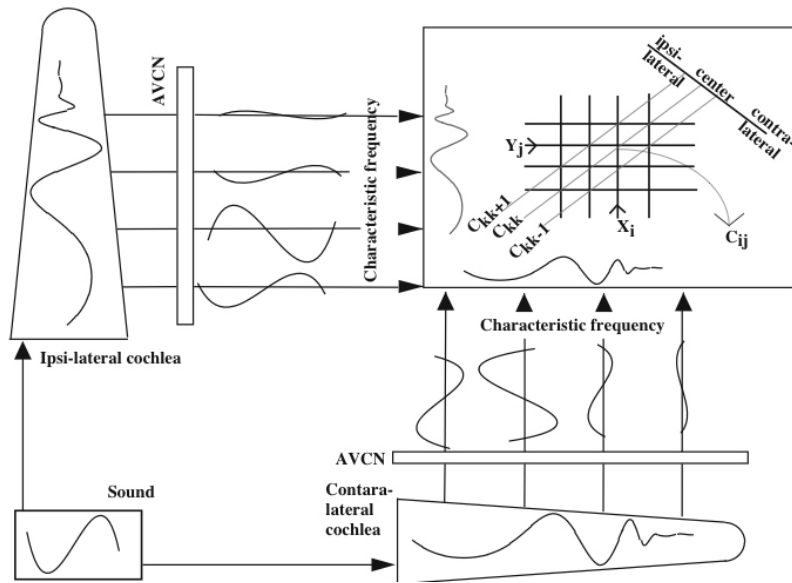


Figure 5.2: The structure of the stereausis network

5.5.1 Stereausis Approach

The stereausis model created by Shamma [SCG89] consists of a so-called *central bin-aural processor*, which is able to process binaural stimuli input. The input signals are preprocessed by two cochlea channels, which reproduces the transfer function of the basilar membrane by modelling it as an *Infinite Impulse Response (IIR)* digital filter. Interaural differences are measured by detecting spatial disparities between instantaneous inputs from the two ears. Without the use of neural delays, the network is able to generate attributes of binaural hearing as the lateralization of sound of all frequencies, as well as the detection and enhancement of noisy signals. This is accomplished by systematically comparing the spatiotemporal responses of a tonotopically-ordered array of auditory nerve fibers at various horizontal shifts. The idea is to perform a cross-correlation between the ipsilateral input at a given characteristic frequency (CF) with the contralateral inputs from locally off-CF locations. The network is therefore able to utilize delays already present in the traveling waves of the basilar membrane to extract correlations.

The stereausis network receives two spatial images (or snapshots) of the traveling waves, one from each ear. If the tone is centered, the images are identical. For unequal signals, the traveling waves differ systematically. When one tone is phase shifted (or delayed) in one ear relative to the other, the instantaneous images appear correspondingly shifted in space.

5.6 Sensor fusion

In the sensor network frame we have at disposal multiple microphone arrays operating simultaneously. The advantage of integrating the results coming from these arrays

might yield a much better sound localization accuracy. The error is a function of the array geometry and on the background SNR. It is interesting to have at disposal not only the most likely value for the time difference between, but also the likelihood of others TDOAs. The method of producing an array of probabilities is what we call *Spatial Likelihood Function (SLF)* generation. An SLF is a noisy approximation of the posterior likelihood $P(\phi(x)|s_1..s_n)$, where $\phi(x)$ is the event that the sound source is located at position x and $\forall i \in [1..n] : s_i(t)$ is the signal received at sensor i at time t . Generally the computation of the value $P(\phi(x)|s_1..s_n)$ is not possible, so we must approximate it with specific methods. A very simple SLF is the generalized cross-correlation ($W(\omega) = 1$). In the noiseless case and when there are no reverberations, a SLF given by a single sensor will be representative of the spatial locations of the sound sources in the environment. In practical situations, it is necessary to combine the SLF of multiple sensors in order to have a more realistic and accurate overall SLF. These methods are well described in [Aar03].

5.6.1 Beamforming

This is one of the oldest technique used in localization using multi-channel. In this approach, we use spatial cues by designing directional filters, which allows to enhance signals coming from particular directions. These techniques were originally used in radar and sonar systems to focus on a beam coming from a particular direction. One example of beamformer is called the delay-and-add proposed by placing several sensors at regular intervals along a line. Depending on the direction where we want to "point the beam" a delay is introduced in order to provide in-phase signals to all sensors, which are summed together to obtain a spatially filtered signal.

5.7 Implementation

5.7.1 Determination of Minimal Sampling Frequency

Let's see some practical assumptions concerning the theoretical minimal sampling frequency to use in order to be able to discriminate a change in time delay equivalent to a difference of azimuth of 1. We used the method proposed by [JAR⁺04] and we had to modify it in order to fit to our convention. Let us consider a variation of the time delay produced by a variation of $\Delta\theta$ degrees at a given angle θ

$$\Delta D = D_{MAX} \cos(\theta) \Delta\theta \quad (5.29)$$

Now if we consider the time delay variation corresponding to a variation of one degree, i.e. $\Delta\theta = 1^\circ = \frac{\pi}{180}$ rad, we get

$$\Delta D_1 = \frac{D_{MAX} \cos(\theta) \pi}{180} \quad (5.30)$$

This equation tells us that to discriminate one degree at an angle of θ , we must sample at a smaller period than ΔD_1 in order to perceive this difference in time delay. The value of ΔD_1 in function of the angle of arrival θ between 2 notes separated by distance of 1.7 m ($T_D = 5000$ ms) can be seen in 5.3, where we notice and notice that the time delay is above 50 μ s for $\theta \in [0, 55] \text{intersect} [125, 180]$. Unfortunately

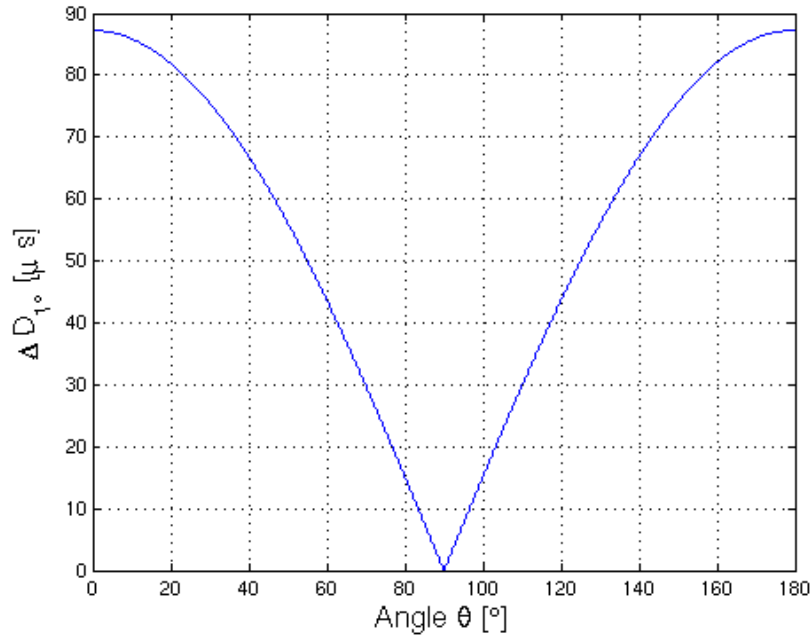


Figure 5.3: The value of the time delay change for a variation of 1 for different values of the angle of arrival θ

such time differences can be perceived only with a sampling frequency above 20 kHz. Interpolation can be carried out easily on the data, but as we sample only at 500 Hz, interpolation to 20 kHz can lead very poor results.

The method we implemented in this project is quite simple. We create a Matlab object for all pairs that can be formed by our sensors, if we have n sensors, we get the pairs on the upper (or lower) part of the matrix. We decided to deal only with *binaural*³ signals, so we'll treat only with a set of pairs of sensors. Must we take each possible pairs between all sensors, or should we be more careful and incorporate each sensor only once in a pair. Now how to find which pairs are the more efficient? For that we decided, to give to each pair a mark. This mark will be representative of the "quality" of the estimation the pair could reach. We used a very simple heuristic, which gives a better mark as the AOA is nearer to 0 or 180, and we penalize the pairs which yield an AOA of 90 degrees, which means that sound source is at an equal distance from the source.

The way we implemented this process is not at all computationally efficient when a large number of sensors are used, as it's complexity is $\mathcal{O}(n^2)$. However, we assume that at a minimal arbitration algorithm is implemented on the motes and only the most relevant sensors send their data. Ideally, our implementation needs only four sensors to localize the sound source, as we can form two pairs each yielding an angle of arrival at the midpoint between the sensors and simply intersect them graphically to obtain the angle. We could also use only three sensors, but in this case the same sensor

³refers to what is perceived at the two ears.

will be used in two pairs, so the results given by the two pairs are not statistically independent anymore. A problem still unsolved is that a single pairs can give only the angle of arrival, but is not able to deduce from which side does the sound come. Thus we can have between 0 and 4 intersection for two pairs. A way to statistically ensure which of them is the true estimate, is to plot all intersections yielded by all possible pairs and check which cluster has the smallest variance.

Another important aspect is that with n sensors we can build up to $\sum_{i=1}^{n-1} i$ different pairs, so how to choose the two most relevant of them? To answer such a question we created a metric assessing the individual performance of each pair. Each pair is given a mark M which is defined as follows :

$$M = \alpha \left(\frac{\pi}{2} - \theta_i \right) + \beta \cdot \frac{\rho_{ij}}{d_{ij}} + \gamma \cdot \left(\frac{2 - p_i - p_j}{2} \right) \quad (5.31)$$

Where α, β, γ are parameters, d_{ij} is the distance between mote i and mote j , and p_i, p_j their respective packet loss percentage.

It would be efficient to also take into account the distance (at least the estimated one) from sensors to the sound source, as the accuracy decreases as the source with increasing distance, but the estimation quality also decreases with the ρ/a (free-field assumption) ratio. It would be interesting to also integrate the quality of the estimation that is the value yielded by the normalized cross-correlation.

Chapter 6

Results and discussions

6.1 Experiments

The first experiment we did was simply to determine the *Quality of Service* (QOS) provided by the motes. The results we get were unexpectedly bad c.f. Figure 6.1, as we get a very high rate of packet loss compared to what we imagined. You can see the mean and deviation of packet loss when we increase the number of motes emitting in the same time to the base-station. The experiment consisted for each mote, to start and keep sampling the microphone and send all data during 5 seconds upon the start message arrival from the base station. Each experiment was repeated 5 times. In the first case the sampling frequency was $f_s = 100$ Hz and we send a packet as soon as we read 10 values. Thus we send 10 packets per second. In the second case we use the same algorithm but with a frequency $f_s = 500$ Hz, and in the third case we sample at $f_s = 1$ kHz. In this algorithm, all mote start to send their packets at the same time. Then we modified the 500 Hz version in order to make each mote send its packet asynchronously, that means each mote starts to send its packets at random times uniformly distributed within ± 15 ms, as the send delay takes less than 1 ms, we thought to reduce the packet collision rate, but we get no significant improvement at all.

We also noticed some other problems, the first being that the base station randomly crashes when 3 or more motes are emitting in the same time, and we had to unplug the alimentation cable to reboot the base station. That was very annoying for performing the tests, as it crashed almost every 5 seconds when we used more than 3 motes simultaneously. We didn't figure out why this happened.

Another interesting thing to note, as all packets are numbered according to the internal timer, we noticed that when we sample at 1 kHz, there's congestion in the scheduling, which requires 11 ms to send each packet with 10 samples, so we lose one sample per packet just for the other tasks to be processed by the CPU, this problem is less significant when sampling at 500 Hz (we lose one ms every 50 ms seconds) and almost inexistent when sampling at lower frequencies such as 200 Hz. This experiment clearly shows that the default scheduler that comes with TOS is not able to efficiently support hard-real time constraints. A support this statement is that we notice in the 1 kHz case that some packets are just "forgotten" and are discarded because the new packet to send is ready while the old one is still waiting, so the priority goes to send the new and the old one is simply discarded. This can be supported by the non-null

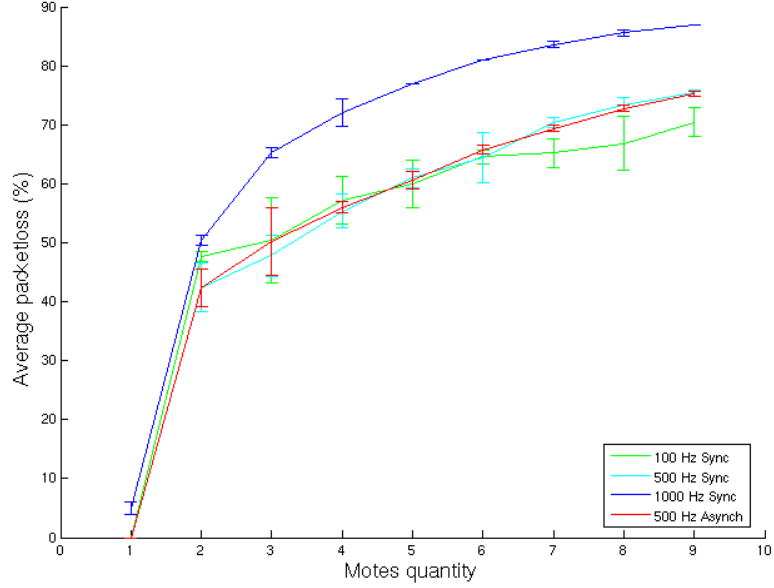


Figure 6.1: The average packetloss percentage for multiple (from 1 to 9) motes emitting in the same time, at different sampling frequencies.

packet loss even when a single mote is emitting, the especially higher packetloss rate compared to 100 Hz or 500 Hz case. The irregular blinking of the led also provides a empiric cue towards this direction.

The datasheet of the MICAZ motes claim that motes should be spaced by at least 1 meter in order to avoid interferences between their oscillators. We tried with much higher distance as well as lower (from 30 cm to 5 m), but this did not influence the packetloss rate. The theoretical maximal number of packets that can be sent per second is around 50 as stated by the manufacturers. This is similar to what the value we get in our experiments on the packetloss.

We perform now some calculations demonstrating the best quality we can reach. We know the sampling frequency f_s , the distance from the midpoint between two sensors to the sound source is denoted ρ , and the angle of arrival is denoted θ .

The first thing is to calculate the maximal time delay τ_{max} between two sensors by a simple model which takes into account according to the sound speed c and the distance between the sensors a :

$$\tau_{max} = \frac{a}{c} \quad (6.1)$$

Now we convert it order to find the maximal delay into samples :

$$d_{max} = \tau_{max} \cdot f_s \quad (6.2)$$

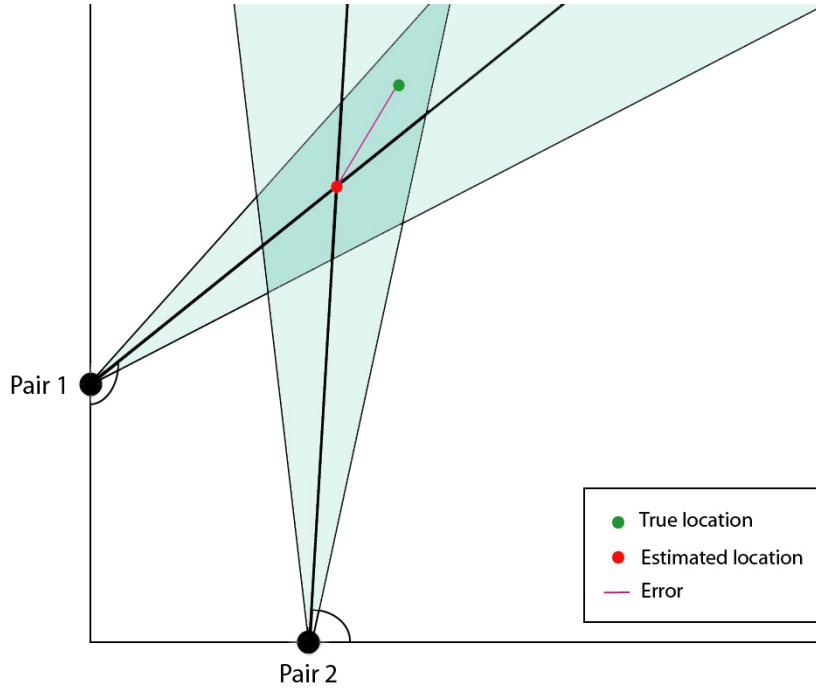


Figure 6.2: The location estimate by intersection of the bearing yielded by two pairs of notes. The blue areas are the cone of confusion of each pair.

that will be used for the cross-correlation as there is useless to compute it for sample delays greater than d_{max} . Now that means we have a discrete case where the azimuth can be given by $2d_{max} + 1$ discrete sensory possibilities, as the real time delay $\tau \in [-d_{max}..d_{max}]$. Where $\tau = -d_{max}$ means the sound source is on the line between the sensors and is located on the left side, $\tau = d_{max}$. When $\tau = 0$, the source is located at the same distance from both sensors. Now the problem is that as we also have $2d_{max} + 1$ different azimuths (note the fact that we don't have enough data in order to know from which "side" does the wave come), and the precision will depend on this factor. If this "azimuthal resolution" is low it will be only a *cone of confusion* approximating the true direction.

The aperture of the cone in degrees can be expressed as follows

$$\theta_e = \frac{360}{d_{max} \cdot 4} = \frac{90}{d_{max}} \quad (6.3)$$

For two sensors ($f_s = 1$ kHz) placed at 1 meter from each other the cone of confusion will have an angle of 30° .

As explained, due to the relatively poor time resolution we have with sensor nodes, we the accuracy we have is quite ridiculous. However we although carried some experiments in order to show that our code works. Sometimes it worked sometimes not depending on the sound used and on the amount of packet loss.

We carried out many different tests, but the problems we had with packet loss

as explained in the next section, made the cross-correlation function yield different results at each trial. For that reason we concluded not to make performance analysis, as the result won't be relevant unless we do a large quantity of tests and average the results. Also, small improvements in the localization accuracy won't appear on the graphs due to the very low accuracy of the angle.

For that reason we were not able to fully test the Matlab code and some bugs might still persist in the package.

Chapter 7

Conclusion and Future Work

As stated throughout this whole project, sensors networks and their applications require many different disciplines as electronics, networking, life sciences. The current state-of-the-art devices are already functional, depending on the application they are even too powerful. But sensing temperature once each 5 minutes is quite different from accurately localize a sound source in real-time. Complex application require all we have at disposal within a computer in a cubic inch volume and that has an extensive autonomy. There's a lot of work ahead. But we can already drastically improve the performances we have with the current technologies, simply by designing each mote for a specific application and using well-known optimizations, such as data compression, efficient OS scheduler or even add a second processor which is dedicated too handle RF communication only thus leaving the main processor able to respond to other real-time tasks. Another interesting with WSN, would be the future technologies of passive sensor devices, for example microphone automatically asynchronously activated upon sound activity detection, without any need for energy expense for periodic active monitoring when there's nothing to hear. In the same way, it would be very interesting to design an extension board dedicated to sound localization with multiple microphones on it. This will allow to perform sampling and correlate the signals in time very precisely. Moreover with the enhanced computation capabilities that will posses the future motes generation, we could directly compute on board the angle of arrival of a sound without the need to defer communication to a remote computer, but only send a single packet per second containing just this information. Then we can simply triangulize and find the intersections of two angles given by two motes and we get the location. This board could integrate a thermistor and a hygrometer as temperature and humidity are the most important factors influencing the sound speed and maybe a dedicated DsPIC for all signal processing tasks. If two microphones are used, depending on the angle of sound source the approximation can be bad if the angle is around 90, but if we use 4 micropones placed in a cross, if the angle is unappropriate, we can switch directly to the other pair. This board should also need some memory to store the sampled signals needed by the cross-correlation.

Another interesting function would be integration an intelligent path planner module, which models the possible paths the sound source will take, so the police just needs to wait the thieves at the place where they planned to get out. We can even imagine coupling a sensor network with mobile robots which can communicate directly via ZigBee protocol and allowing robots to query specific information about the environement as it needs enabling them exteroception, or in the contrary the

sensor network could directly ask robots to act upon the environment for moving cameras into specific locations and so on.

Concerning our application, we plan to further develop it in order to implement real-time localization upon packet reception from the network while continually investigate into potentially more accurate localization techniques.

We discussed the main aspects concerning the implementation a low-power algorithm for sound localization using a distributed sensor network. We showed that the current COTS sensor nodes are not powerful enough in order to perform efficiently sound localization using very low energy and simultaneously yield precise results. However, the Matlab framework we developed during this project is not limited by all the constraints inherent into sensor networks, and is able to perform very precise localization according to the sampling frequency of the signals. It has the ability to contain a large quantity of time-correlated signals, plot the individual location estimates yielded by all pairs of sensors, and finally make a statistical (RMS) average of all possible source location and yield a final value. There is still a lot of work to do in miniaturization of sensors network hardware devices and in the sound localization models, which take into account the variety and complexity of natural sounds as the human auditory system does. It is only after a rigorous and methodical process of alternating series of designs, tests and verifications that the full potentials of sensor networks can be realized.

The next generation of sensor nodes with increased processing power and storage, low power MEMS (sensors, radio modules) with much smaller physical sizes, will give birth to a totally new class of devices and will definitely revolutionate our lives. The progress made by core sciences, especially nanotechnology will provide us in a very near future almost invisible year-lasting autonomy devices and their potential connection to all this data could give birth to a novel type of entity, a world-scale organism that will see and hear everything, smell the most hidden part of each factory, and spotlight the darkest zone in the world. Combine this yet unseen universal sensing capability to the biggest knowledge database upon our world, and finally connect the whole to zillions of activators located everywhere, from each supermarket doors, to elevators, passing through air climatization, circulation regulators, DoD military installations and warfare. An entity that will see, know and act upon all of us. This strong analogy was also stated by Albert Laszlo Barabasi in his world famous best-seller, *Linked*. As stated by the french metteur en scene Louis Juvet : "Les hommes ont cree les dieux a leur image, l'inverse reste a prouver."¹. Some ethical questions should arise at this point.

¹The humans created the gods to their image, the contraposal of this statement still remains to be proved.

Appendix A

7.1 How to use TOS

We assume that you have a functional installation of TinyOS, a MIB510 programming board attached through the COM1 serial port to a base computer running a Linux operating system. We also assume that the MICAZ motes described in the chapter 2 are used.

7.1.1 Compile a program

You can compile directly a program using the provided Makefile, just by typing in the directory where is located the program to compile :

```
make micaz
```

This command will only compile locally the code on the computer without effectively installing it onto the destination mote attached to the programming board. To install it physically to the device, you must use this command

```
make install.<address> micaz
```

where **address** is the desired local address the mote will receive. Note that the compiler uses the environment variable **MIB510** to store the information on how to access the programming board, in our case the file associated with the COM1 serial port, that is the file `/dev/ttyS0`.

We also need to specify in the makefile which sensor board is used in order to link the specific libraries provided with the board when compiling the application. As we use the MTS300CA board, we need to specify the following line in the Makefile in order to use the libraries :

```
SENSORBOARD=micasb
```

Every mote also has a Group ID, in addition of its local address. That allows to use multiple node group, without interfering packets and/or avoiding to use packets coming from an unknown source. The default GID is `0x7d`.

7.1.2 Data retrieval

The mainly used application used in retrieval of the packets coming from the network is the SerialForwarder application. A simpler version, is the Listen program which also displays the packet content into the terminal.

All these programs use the **MOTECOM** environment variable which tells where to read packets. this variable must be set to :

```
export MOTECOM=serial@COM1:57600
```

The baudrate used to communicate with the MICAZ motes is 57600, while for example, the MICA2DOT use 19200. For this purpose, you must install the TOSBase application on the mote plugged on the programming board, its purpose is simply to forward data between radio and UART bidirectionally.

Appendix B

7.2 The Java Localizer Program

This program derives from the `Oscilloscope` example that comes with TinyOS, but has been modified in order to handle the packets coming from the network and the transform the data they contain into sound files, in fact that are simple Matlab scripts that contain relevant data about the experiment. As this part was not the most important, we neglected the user interface but we focused on the utilization. We plan to remove this part in the future version of the library and be able to catch the packets directly in Matlab in order to perform localization in real-time without the need for an intermediate program. You can launch the program by copying the project directory into `/$TOS/tools/java/net/tinyos`, where `$TOS` is your TinyOS installation directory. Then go to the `/$TOS/tools/java` directory and type :

```
user@computer$ java net.tinyos.project.localizer &
```

Assuming that you have your base station running the `TOSBase` application correctly plugged on the serial port, and the `TOS` environment variable configured as explained in the previous chapter, a window similar to 7.1 should appear.

The only important button is this program id the start/stop sync used to record sounds. After pressing this button, a sync message will be emitted and will synchronize all motes, who will reset their counters sleep for 1 second, and start monitoring the environment. Every packet send within the 4 following seconds will be recorded by the program and you will be asked the name of the file where you want the data to be stored. It is not impossible that one could repeat this process once or twice, due to some internal synchronization bug. Then you are ready to process the file you just created, as will be explained in the next section.

7.2.1 Sound Files

We use these files as an intermediate representation of the sensor readings for a specific event. These files are simple scripts that can be run into Matlab and an example is shown in Fig. 7.2. It contains a `mote(:,i)=[n,s(1)..s(t)]` entry per mote, where n is the number of the mote (and not $i!$) and $s(j)$ are the reading of the at time j , correlated with the samples from other motes. Then the `p(n)` is the packetloss in % for mote n . `env` is the environment file containing the configuration of the experiment. `Freq` is the sampling frequency f_s and we assume that all motes sample at the same frequency. Finally `sourcex`, `sourcey` are the exact coordinates of the sound relative to the environment of this experiment.

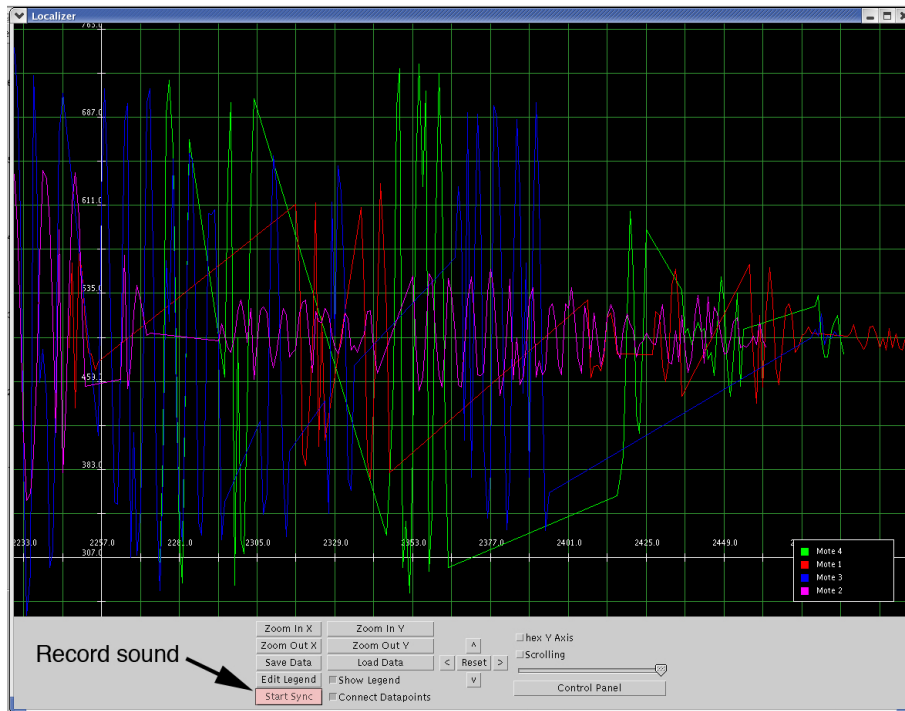


Figure 7.1: Interface of the java localizer program used to transform the packets from the network into Matlab scripts containing the sensor readings correlated in time. Long diagonals denote packetloss.

```
% mote 3 was on channel 0
mote(:,1) = [3,477, 520, 479, 508, 464, 500, 504, 474, 503, 507]
% mote 2 was on channel 1
mote(:,2) = [2,512, 512, 512, 512, 512, 512, 470, 517, 481, 494]
% mote 1 was on channel 2
mote(:,3) = [1,512, 512, 512, 512, 512, 512, 512, 512, 505, 463]
% Packet Loss Stats
p(1) = 54;
p(2) = 45;
p(3) = 57;
env = 'swis3.env';
freq = 500;
sourcecx = 374;
sourcecy = 505;
```

Figure 7.2: Example of a sound file generated by the Localizer program.

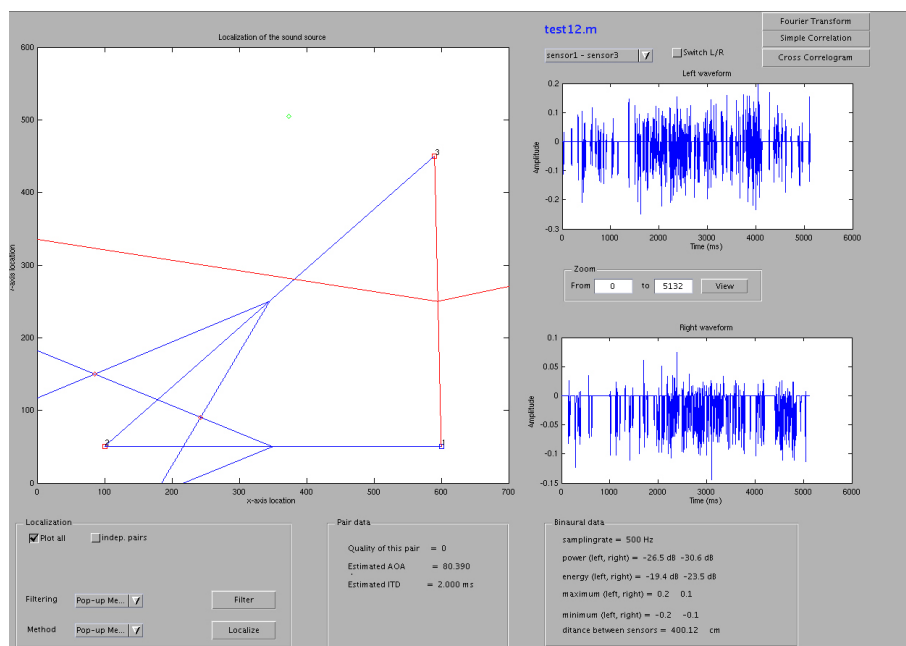


Figure 7.3: Graphical interface of the sound localization library.

7.3 The Matlab GUI

We developed during this project a package, containing a general purpose framework for sound processing under Matlab. This software is still in its early stages of development, but the modular skeleton is already functional.

Start Matlab from the directory `locscripts` where are the Matlab scripts located. Then just type `gui` in the Matlab command prompt. A big empty window with just a file menu will open.

You can open a sound file generated with the java localizer by selecting `Load sound...` in the file menu. Before that you must ensure that the environment file for this experiment is in the `locscripts` directory. Once opened a lot of new things will appear as shown by Figure 7.3.

On the right side of the screen you can see the binaural signals plotted and between the graphs you can find a panel where you can select which part of the signal you want to focus on, as shown in Figure 7.4.

Over these plots are located the name (in blue) of the sound event file being currently processed. Under the name is a popup menu where you can select which pairs you want to focus on. The three buttons show some plots concerning the signals, such as their cross correlation of Fourier transform, and a checkbox allowin to invert left and right channel. Note that this option has no effect on the localization process. You can see these commands in Figure 7.5

Under the plots, we find 3 panels which are intended to be customized by the user in order to fit their needs. The left panel shown in Figure 7.6 contains commands concerning the sound source location as menu for selecting a filter to apply to all

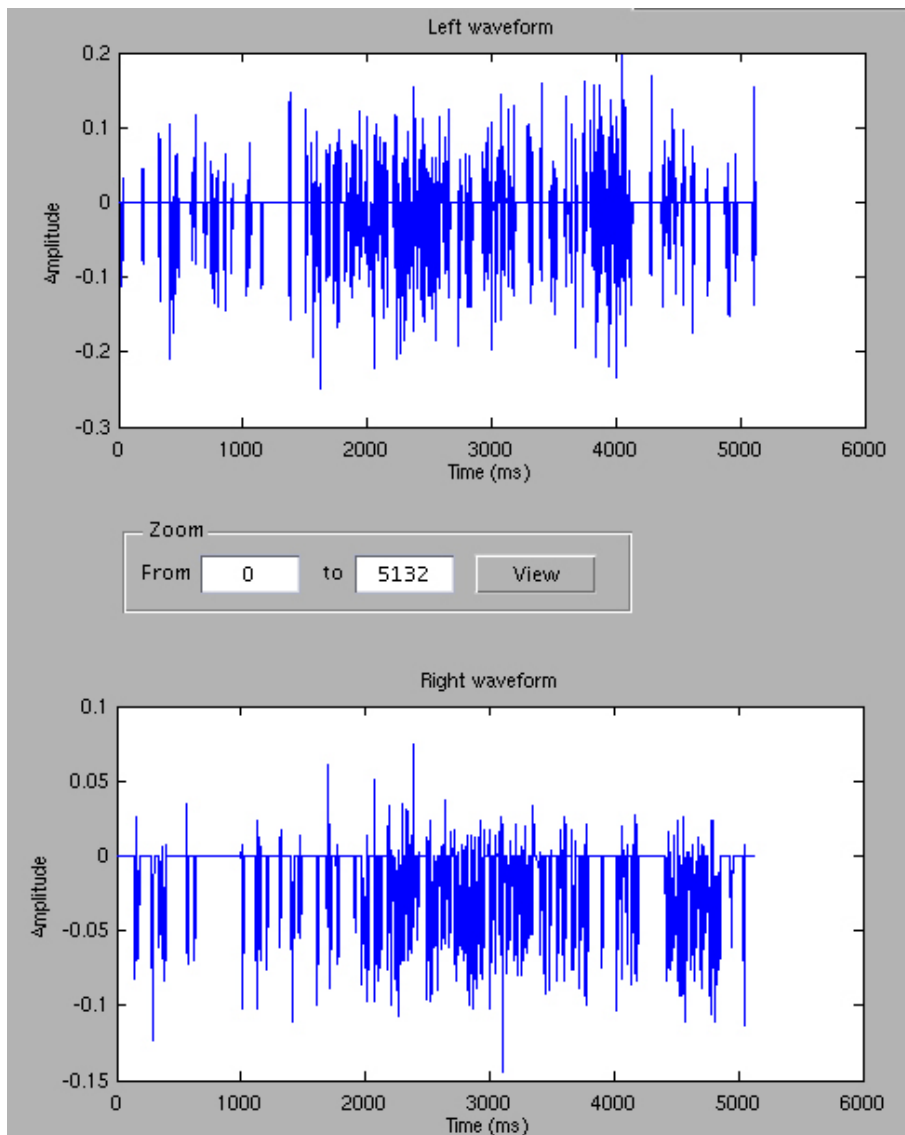


Figure 7.4: Binaural signals.

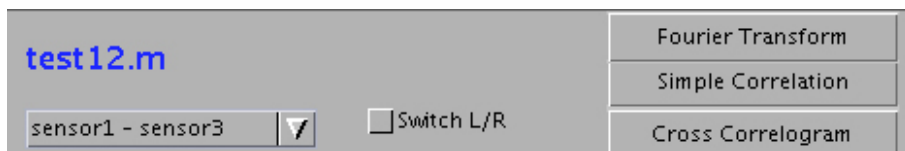


Figure 7.5: General command for the sound signals processing and pair selection.

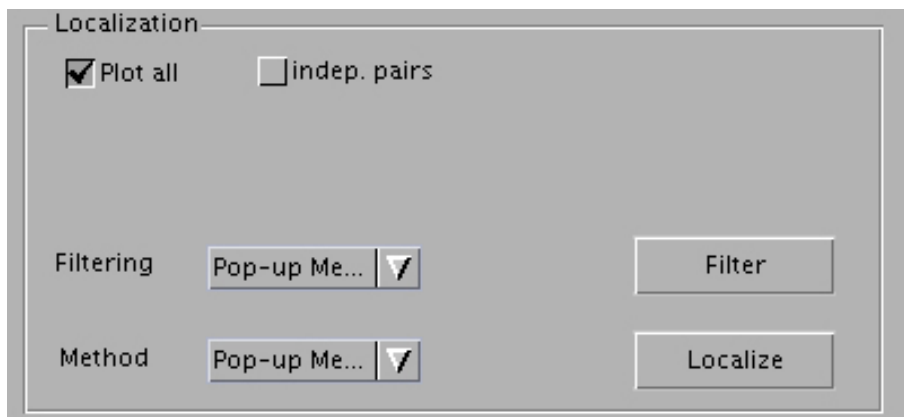


Figure 7.6: Command panel for filtering and localization of sounds.

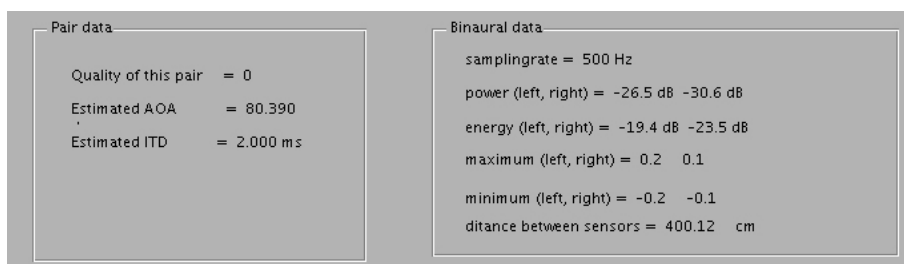


Figure 7.7: Panels containing information concerning binaural signals.

signals, and what method to use to localize the sound. In the middle, the panel shown in Figure 7.7 contains statistics concerning the pair as the mark of this sensor pair and the ITD estimation for this pair and the angle of arrival corresponding to this ITD.

7.3.1 Environments

Each environment is described in a file that has a `.env` extension and strictly follows the syntax as shown in Figure 7.8. On the first line you must give the name of an image file that is the map of the environment. This feature is not yet used in our current implementation. On the second and third line you can specify the size in cm of the environment. Then you can enter a line for each mote using this syntax `NUM XPOS YPOS`, where `NUM` is the mote number, and `XPOS YPOS` are the mote coordinates relative to this environment. Position `(0,0)` is located in lower-left corner on the map in the matlab script.

```
IMAGE 'swis.jpg'  
HEIGHT 600  
WIDTH 700  
1 600 50  
2 100 50  
3 600 450
```

Figure 7.8: Example of the content of the configuration file for a given environment.

Bibliography

- [Aar03] Parham Aarabi. The Fusion of Distributed Microphone Arrays For Sound Localization. In *EURASIP Journal of Applied Signal Processing (Special Issue on Sensor Networks)*, pages 338–347, March 2003.
- [AKLM04] T. Ajdler, I. Kozintsev, R. Lienhart, and M.Vetterli. Acoustic Source Localization in Distributed Sensor Networks. In *Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, 2004.
- [Ayl03] D. Aylor. Noise Reduction by Vegetation and Ground. In *Journal of the Acoustical Society of America*, page 197, 2003.
- [BCP02] Mark F. Bear, Barry W. Connors, and Michael A. Paradiso. *Neuroscience, Exploring the Brain*. Lippincott Williams and Wilkins, 2002.
- [BRS] Ehud Ben-Reuven and Yoram Singer. Discriminative Binaural Sound Localization.
- [CHE02] M. Cagalj, J. P. Hubaux, and C. Enz. Minimum energy broadcast in all-wireless networks : NP-completeness and distribution issues. In *Proceedings of ACM MobiCom*, pages 178–182, Atlanta, Georgia, September 2002.
- [CYE+03] Joe C. Chen, Len Yip, Jeremy Elson, Hanbiao Wang, Daniela Maniezzo, Ralph E. Hudson, and Kung Yao. Coherent Acoustic Array Processing and Localization on Wireless Sensor Networks. In *Proceedings of the IEEE*, pages 1154–1162, August 2003.
- [Fel03] Carlos Felgueiras. Power Spectrum Estimations, FFT methods and Coarse Grained Spectral Analysis. 2003.
- [GKvHW96] W. Gerstner, R. Kempter, J.L. van Hemmen, and H. Wagner. A neuronal learning rule for sub-millisecond temporal coding. In *Nature*, pages 76–78, 1996.
- [GS01] E. Grassi and S. A. Shamma. A Biologically Inspired, Learning, Sound Localization Algorithm. In *Conference on Information Sciences and Systems*, 2001.
- [GW] Sandeep K.S. Gupta and Bin Wang. Energy-Efficient Multicast Protocols. In *Resource Management in Wireless Networking*.

- [HSI⁺01] John Heidemann, Fabio Silva, Chalermek Intanagonwivat, Ramesh Govindan, Deborah Estrin, and Deepak Ganesan. Building Efficient Wireless Sensor Networks with Low-Level Naming. In *Symposium on Operating Systems Principles*, pages 146–159, 2001.
- [HSW⁺00] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System Architecture Directions for Networked Sensors. April 2000.
- [JAR⁺04] P. Julian, A. G. Andreou, L. Riddle, S. Shamma, D. H. Goldberg, and G. Cauwenbergs. A Comparative Study of Sound Localization Algorithms for Energy Aware Sensor Network Nodes. In *IEEE Proceedings on Circuits and Systems*, pages 640–648, April 2004.
- [JCCS04] Craig T. Jin, Anna Corderoy, Simon Carlile, and Andre Van Schaik. Spectral Cues in Human Sound Localization. In *Journal of the Acoustical Society of America*, pages 3124–41, June 2004.
- [Jef48] L. A. Jeffress. A Place Theory of Sound Localization. In *J. Comp. Physiol. Psychol.*, pages 35–39, 1948.
- [KC76] Charles H. Knapp and G. Clifford Carter. The generalized Correlation Method for Estimation of Time Delay. In *IEEE Proceedings on Acoustics, Speech and Signal Processing*, pages 320–327, August 1976.
- [KC90] M. Konishi and C.E. Carr. A Circuit for Detection of Interaural Time Differences in the Auditory Stem of the Barn Owl. In *J. Neuroscience*, pages 3227–3246, 1990.
- [Kon73] M. Konishi. How the owl tracks its prey. In *Am. Sci.*, pages 414–429, 1973.
- [Lew04] F.L. Lewis. Wireless Sensor Networks. In *Smart Environments: Technologies, Protocols and Applications*, New York, USA, 2004.
- [MG03] David McAlpine and Benedikt Grothe. Sound localization and delay lines – do mammals fit the model? In *Trend in Neuroscience*, pages 346–350, July 2003.
- [MPS⁺02] Alan Mainwaring, Josep Polastre, Robert Szewczyk, David Culler, and John Anderson. Wireless Sensor Networks for Habitat Monitoring. In *WSNA 02*, Atlanta, Georgia, USA, 2002.
- [NBA] Dibyendu Nandy and Jezekiel Ben-Arie. A Comparison of Auditory Localization Models.
- [Pu98] Chiang-Jung Pu. *A Neuromorphic Microphone for Sound Localization*. PhD thesis, University of Florida, 1998.
- [Ray07] Lord Rayleigh. On our Perception of Sound Direction. In *Phyl. Mag.*, pages 214–233, 1907.
- [RS00] Jason Riedy and Robert Szewczyk. Power and Control in Networked Sensors. 2000.

- [SB00] Suanyana Saha and Peter Bajcsy. System Design Issues in Single-Hop Wireless Sensor Networks. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, April 2000.
- [SCG89] S. Shamma, N. Chen, and P. Gopaldaswamy. Stereausis : Binaural processing without neural delays. In *Journal of the Acoustical Society of America*, pages 989–1006, 1989.
- [VE03] Harald Viste and Gianpaolo Evangelista. On the Use of Spatial Cues to Improve Binaural Source Separation. In *Proceedings of 6th International Conference on Digital Audio Effects (DAFx-03)*, London, UK, 2003.
- [VE04] Harald Viste and Gianpaolo Evangelista. Binaural Source Localization. In *7th International Conference on Digital Audio Effects (DAFx-04)*, Naples, Italy, 2004.
- [Vis04] Harald Viste. *Binaural Source Localization*. PhD thesis, Ecole Polytechnique Fédérale de Lausanne, 2004.