DEVELOPMENT OF A HIGH FREQUENCY AMBIENT NOISE DATA ACQUISITION SYSTEM

KOAY TEONG BENG

NATIONAL UNIVERSITY OF SINGAPORE

2004

DEVELOPMENT OF A HIGH FREQUENCY AMBIENT NOISE DATA ACQUISITION SYSTEM

KOAY TEONG BENG

(B.Eng.(Hons.) UTM)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2004

Name: Koay Teong Beng

Degree: M. Eng.

Department: Electrical and Computer Engineering

Thesis Title: Development of a high frequency ambient noise data acquisition system

ABSTRACT

High frequency ambient noise has a significant impact on the operation of many Sonars and related systems. Therefore, understanding the temporal and spatial distributions of this noise in shallow water is crucial. Existing high-bandwidth acoustic data acquisition systems are large and complex. This project has developed a novel stand-alone, portable, and compact cylindrical package (23cm ∅ by 60cm length) that can be rapidly and flexibly deployed in various configurations. It has 4 simultaneous sampling analog channels (up to 5MSa/s aggregate) and is capable of beamforming in 3D space using a tetrahedral array configuration. This system has provided both time-space distributions and directivity of high frequency ambient noise in Singapore waters for the first time.

Keywords: High Frequency, Ambient Noise, Snapping Shrimp, Acoustic, TDOA, Shallow Water, Spatial and Temporal Distribution

ACKNOWLEDGEMENTS

I would like to acknowledge the Acoustic Research Laboratory of the Tropical Marine Science Institute for their support throughout the project, especially A/Prof John Potter for his continuous guidance and patience over the years. Also, this project would not be possible without the dive team support, they are: Mr. Eric Delory, Mr. Sorin Badiu, Mrs. Caroline Durville, Mr. Mandar Chitre, Dr. Matthias Hoffmann-Kuhnt, and A/Prof John Potter.

In addition, I would also like to conduct my appreciation to Dr Venugopalan Pallayil and Mr. Mohanan Panayamadam for the administrative and field trip support, Mr Mandar Chitre and Dr. Matthias Hoffmann-Kuhnt for proof reading the thesis.

This work has been supported by the Singapore DSTA Research Directorate.

TABLE OF CONTENTS

SUMMARY

It is known that high frequency ambient noise level is significantly higher in warm shallow water compared to deep-water ambient noise, which would affect the operations of various underwater equipments. Researchers have found that snapping shrimp noise is the dominant component (around 190dB re 1uPa @ 1m peak to peak) within frequency range from 2kHz to more than 300kHz) in such regions. At the time of this project, there are very few studies of high frequency ambient noise directivity and its source distribution, and none in Singapore. The project aims to fill this research gap.

At the initial stage, a set of remotely controlled (client server based), seabed mounted, directional receivers were developed. When working together, they are capable of mapping the snapping shrimp acoustic sources on the seabed using a stochastic tomography inversion algorithm. As the project evolved, a much more portable, compact and flexible quad channel acoustic array, named High frequency ambient noise Data AcQuisition System (HiDAQ), was developed. It enabled local researchers to rapidly study the ambient noise in waters that are geographically confined. To the best of our knowledge, this is the first system in the world with such mapping capability that supports sampling rates of up to 5MSa/s.

This thesis focuses on the description of the HiDAQ, its development, principle of operation, field trials and results. The system was designed based on industrial technologies and embedded systems. A prototype electronic compass based on a magneto-resistive sensor has also been built; its theory of operation is also discussed. A Time Difference Of Arrival (TDOA) based beamforming algorithm was developed and used in the field experiments.

Data obtained from field trials in the Southern Islands in Singapore are presented in this thesis. This project has collected directivity measurements of local ambient noise and spatial and temporal distributions of local ambient noise source levels in Singapore for the first time.

Two conference papers (student as principle author) have been presented in OCEANS conferences in the year 2002 and 2003 on the system and the snapping shrimp acoustic distribution study. Another two other conference papers related to the usage of HiDAQ (research student as co-author) have been presented in year 2003.

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Many marine related scientific survey systems use acoustics. After the end of the cold war era, marine communities and researchers started to diversify resources from deep-water operations to study acoustics in shallow waters. Recent worldwide terrorist threats have also generated a lot of interest in homeland security for many countries, which has led to increased operations in their local waters. These operations include defending the coastlines against small and yet potentially hostile subjects and water column monitoring in shallow waters. Since shallow waters do not support the use of low frequency acoustics efficiently, high frequency acoustics has been extensively used for its operational feasibility. High frequency sonar is capable of interrogating subjects and the environment in a smaller geometry, which suits the nature of the shallow water environment where objects of interest are generally smaller. Nevertheless, ambient noise in warm shallow waters level is alarmingly stronger (more than 25 dB higher) compared to the deepwater ambient noise [1] and significantly affects the operations of these sonar systems. Therefore, it is crucial to understand the structure of the high frequency ambient noise in order to effectively operate these systems.

Being an island country and one of the busiest ports in the world, Singapore needs to effectively manage its surrounding marine resources, maintain security in its local waters for commercial shipping, protect its high value assets around the coastlines etc. For these reasons, Singapore is continuously monitoring, exploring and studying the marine environment. These efforts involve an extensive use of high frequency oceanographic equipment in local waters. High frequency ambient noise in Singapore waters is dominated by snapping shrimp (genera *Alpheus, Synalpheus & Penaeus*) [2], hence studying the acoustics of these creatures will give us a good understanding of local ambient noise at high frequencies. Although there are many ambient noise studies, there are very few attempts to map the ambient noise sources and no such experiment has been conducted in Singapore waters prior to this project.

Work done in this thesis is aimed at studying the spatial and temporal distribution of ambient noise source in shallow waters [3] and to produce some maps of noise source distribution in Singapore waters for the first time. The results generated from this work support various experiments carried out by the Acoustic Research Laboratory at the Tropical Marine Science Institute. This project involved the development of a robust, portable and easy-to-deploy instrument for the purpose of this study in particular and for the purpose of studying high frequency transients in 3-dimensional space in general. The equipment developed has proved very useful and has contributed to various scientific underwater studies at the laboratory that have involved high frequency acoustics such as bio-sonar, shrimp noise directivity [4], humpback whale acoustics [5] and, recently, living resource classification replacing the system used in the initial attempts [6].

1.1 Background

Acoustics is one of the best and most efficient tools to investigate the aquatic environment. High-frequency Electro-Magnetic (EM) waves do not travel far in seawater due to attenuation (about 18dB attenuation per meter at 180kHz in seawater), limiting its to short range operations or the usage of very low frequency range (hence a large antenna) for long range operations [7]. These factors make it an unattractive choice to be used underwater. Laser systems have been used in various areas for short-range applications, where the operation ranges are highly dependent on the turbidity of the medium. Acoustic energy, on the other hand, travels efficiently in seawater and is widely used in modern underwater systems for various applications such as geoacoustic studies, bathymetry studies, navigation, communication etc.

Snapping Shrimp produce high-energy broadband noise through the collapse of cavitation bubbles [8]. They are known to dominate shallow water ambient noise from 2kHz to over 300kHz [9], at peak-to-peak source levels of 190dB re 1 uPa @ 1m [10]. These transients could present severe interference to many sonars and need to be suppressed with various transient suppression techniques. On the other hand, ambient noise can also be used as a tool for imaging and the Acoustic Research Laboratory of the Tropical

Marine Science Institute at the National University of Singapore has developed a next generation sonar system, named ROMANIS, that uses these signals to create an acoustic image of the environment [11].

Therefore, understanding the temporal and spatial distributions of high frequency ambient noise sources is one of the key factors for sonar operators to efficiently operate a high frequency system. These problems have lead to a number of ambient noise studies in shallow waters, some examples are [9] [12] [13] [14]. Nevertheless, there are limited studies in Singapore.

Acoustic Research Laboratory has conducted a series of ambient noise studies in Singapore waters using an omni-directional acoustic recording system [9]. The results revealed that the probability distribution function of the ambient noise power exhibits an approximately lognormal distribution. This suggested that the ambient noise sources seem to cluster either in time or space, or possibly both [15]. In order to explain the distribution, the spatial and temporal distributions of these noise sources need to be mapped. There were only a few such experiments in the world, which normally involved large structures and arrays [16] [17]. Furthermore, these projects looked at frequency ranges below 100kHz

The aim of this project is to produce the spatial and temporal distributions of high-frequency underwater ambient noise sources for the first time in Singapore waters. This project studied the ambient noise over a large frequency range (from 1kHz up to 200kHz). A robust and portable instrumentation was developed to estimate the angular distribution, range, and source levels of transient sources in three-dimensional space. It is also flexible enough to serve as a multi-purpose, multi-channel high frequency data acquisition system. The directivity of local ambient noise was studied for the first time

A single acoustic array that is compact, portable, and capable of being deployed at open sea was desired. The system needs to estimate the direction of arrival, range, and source levels of transient sources of local ambient noise (dominated by snapping shrimp). This calls for acquisition

hardware with at least 4 acoustic channels, each acquiring signals up to 200kHz, to cover the majority frequency range of snapping shrimp noise. Therefore, we needed a four-element spatial array to sample the acoustic signals in three-dimensional space with at least 400kSa/s per channel to avoid aliasing of the signals. All four channels had to be synchronized to allow for beamforming. Furthermore, the data had to be streamed to storage devices in real-time with a minimum continuous recording speed of 1.6MSa/s.

Commercially available data acquisition systems with such specifications are based on desktop Personal Computers (PC), which are not suitable for this application. Desktop systems are bulky, and not portable; they also need an AC power supply. Most desktop PCs need air ventilation and therefore can't be sealed to work underwater; this also makes them unsuitable to work in sea breezes due to the threat of corrosion to the electronics. An embedded system that runs at low power had to be built to address these issues.

The designed HiDAQ is capable of simultaneously sampling four analog channels with aggregated sampling rates of up to 5MSa/s with 12-bit resolution. The analog channels are connected to four hydrophones with 5-meter long flexible cables, allowing it to be arranged in various array configurations. The system stores the acquired data into a built-in high-speed SCSI harddisk. The directivity of the sources and their distribution map is obtained after post-processing.

In the post processing process, the system deterministically identified high frequency transient in all four channels. Once they are identified, their inter-channel time delay is determined and used to beamform the transient direction. Although the array is sparse, the beamforming is possible because the dominant ambient noise sources are broadband and impulsive in nature. Figure 1 shows pictures of a partial setup: 1) surface mounted and 2) bottom mounted configuration.

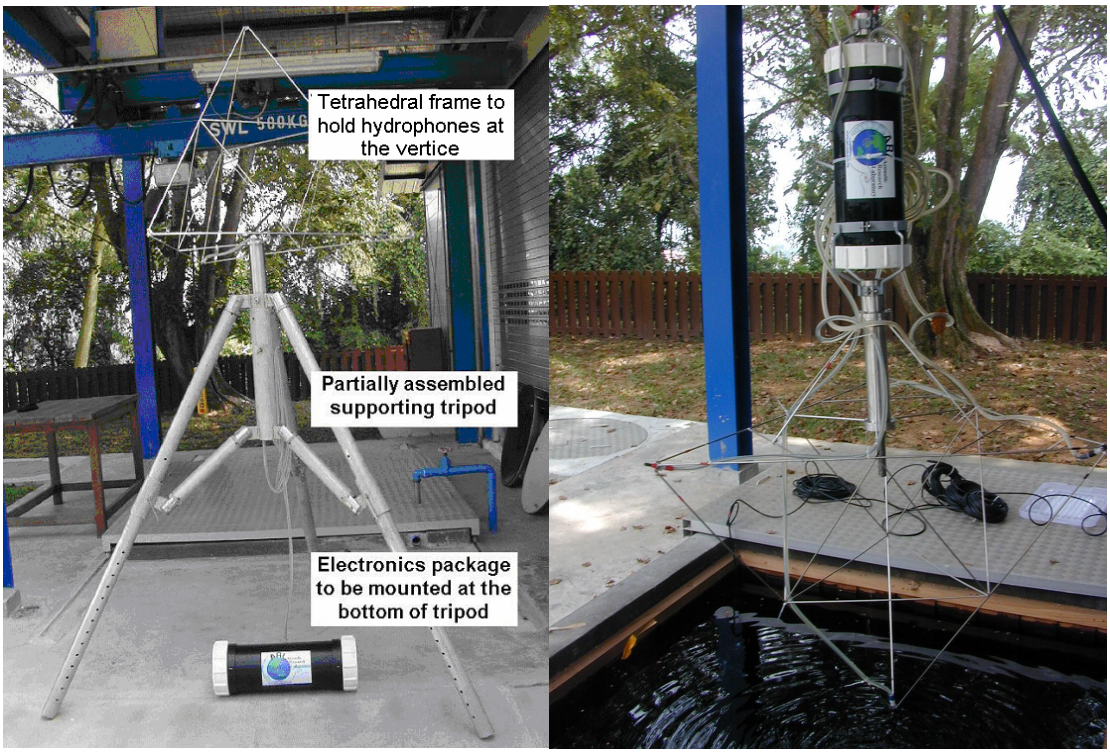


Figure 1: Partial setup of the bottom mounted (left) and surface mounted (right) configuration

CHAPTER 2
SYSTEM DESIGN

HiDAQ was designed using standard industrial form factors, interconnections and communication protocols in order to keep the cost low and to allow for the use of a wide diversity of existing industrial electronic modules. HiDAQ was built from Commercial Off-The-Shelf (COTS) technology with a customized analog front end and signal conditioning circuitries. One of the challenges of the hardware system design was to run the system on a low power CPU using a stripped down version of Embedded Windows NT to conserve power and yet to provide enough CPU resources for the acquisition task. This was done by integrating a COTS data acquisition card and high-speed storage system on a low-power Pentium processor in PC104+ format. The system could either be operated from battery power or from AC power. It also provided a number of human interface modes with the system OS and the acquisition software.

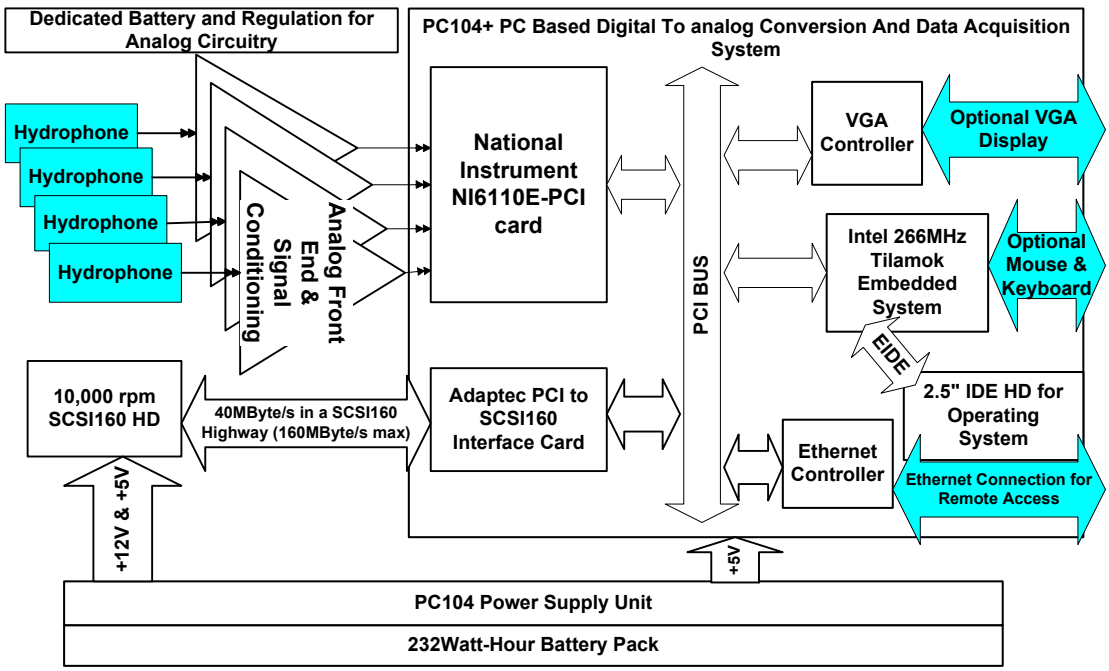


Figure 2: Block diagram of HiDAQ hardware

An embedded CPU system based on the PC104+ form factor was chosen because it provided numerous standard PC peripherals and

communication protocols. A converter board and adapters were used to bridge the PC104+ interconnection formats to standard desktop PC interconnection formats. A high speed Analog-to-Digital Converter (ADC) card in standard PCI edge connection was used as the digitizer, and a PCI SCSI160 adapter was used with a 80GB SCSI harddisk to provide high-speed data storage. A standard 2.5” laptop IDE harddisk was used to store the operating system, thus isolating the data storage harddisk from any delays caused by OS-related accesses. An analog signal conditioning circuitry was designed in-house to receive signals from the four hydrophones, to provide amplification and filtering, and then to feed the signals to the ADC. The system could be powered from one of two power supply options: the first is a Li-Ion battery pack and the second is 230V AC power line through a modified mini-ATX power supply. This made HiDAQ capable of running both as a standalone system for short-term deployment and as a surface powered system for long-term deployment. Figure 2 shows a block diagram of the system. The sections in blue are parts that interconnect the internal electronics to external devices; they include the hydrophones connections and the communication links to the electronics. These parts were packed in a compact mounting cage (see Figure 3), which was then mounted in a cylindrical watertight housing.

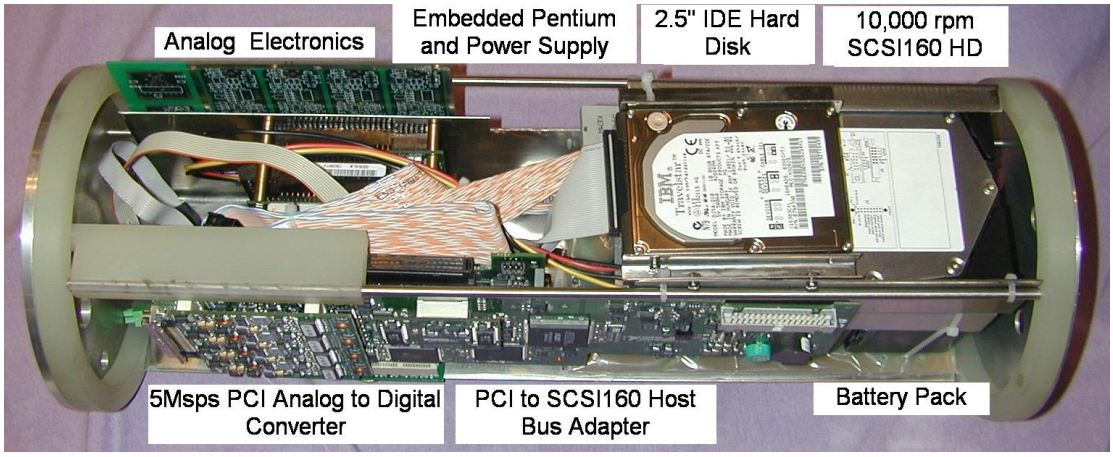


Figure 3: Electronic modules packaged in a compact mounting cage.

The complete electronics package, including watertight housing, is a cylindrical package of less than 23cm in diameter by 60cm in length and weighs about 25kg in air and about 5kg in water. Figure 4 is an illustration of

the electronics package in the housing with hydrophones and mounting brackets attached.

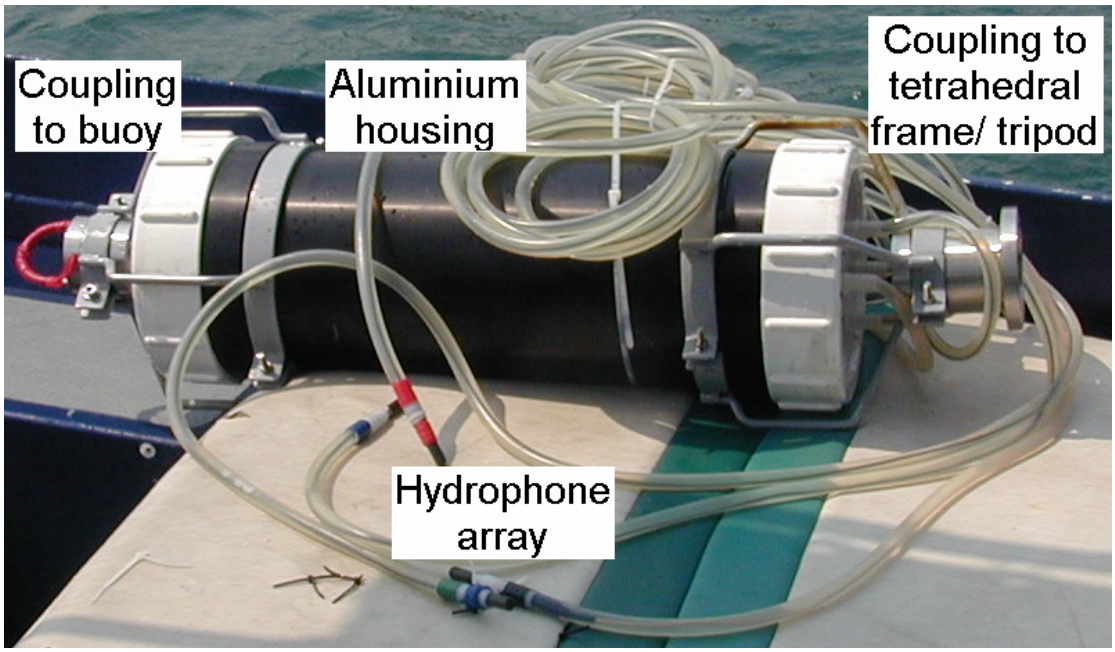


Figure 4: HiDAQ electronics package ready to be deployed

2.1 Embedded Pentium PC

Several Pentium-based embedded processors were evaluated for the purpose of the project. Although high-performance embedded processors (500MHz or higher) would have been desirable for the acquisition system, the heat dissipation problem and the large power consumption made them unsuitable for the project. A processor with moderate processing power was used with a high-end data storage interface and an acquisition card with a good buffer scheme to provide desirable performance. Furthermore, in order to reduce the overhead to the CPU, the operating system was stripped down to the minimum required. Table 1 shows a comparison on the power consumptions and features of some of the embedded PCs considered.

Table 1: Comparisons of performance between some industrial PCs

CPU	AMD SC520	Geode	Pentium Tillamook MMX	Pentium III
Format	PC104+	ETX	PC104+	EBX
Max Speed	133MHz	266MHz	266MHz	750MHz
RAM	64Mbyte	128Mbyte	128Mbyte	512Mbyte
Peripherals	Full PC peripherals	Full PC peripherals	Full PC peripherals	Full PC peripherals
Typical Power consumption	4W	5W	8W	20~25W
VGA controller	No	Yes	Yes	Yes
Ethernet controller	Yes	Yes	Yes	Yes
Processor performance	Low	Medium	Upper medium	High

The Central Processor Unit (CPU) chosen was an industrial embedded PC in the PC104+ platform that was built around Intel's Pentium Tillamook 266MHz MMX processor. This system was chosen for its low power consumption, its compatibility to Windows NT, and its ability to provide a complete range of PC peripherals. Furthermore, the system included a built-in VGA controller and Ethernet controller with transceiver. It was installed with 128Mbyte of SODIMM SDRAM, providing enough memory space for the acquisition application software and advanced buffering for the ADC operations. The embedded PC was built around standard electronic components used in desktop motherboards; therefore it was supported by the widely available device drivers for standard operating systems. In addition, it also guaranteed good inter-operatability with other standard industrial modules. Figure 5 shows the PC104+ module used.

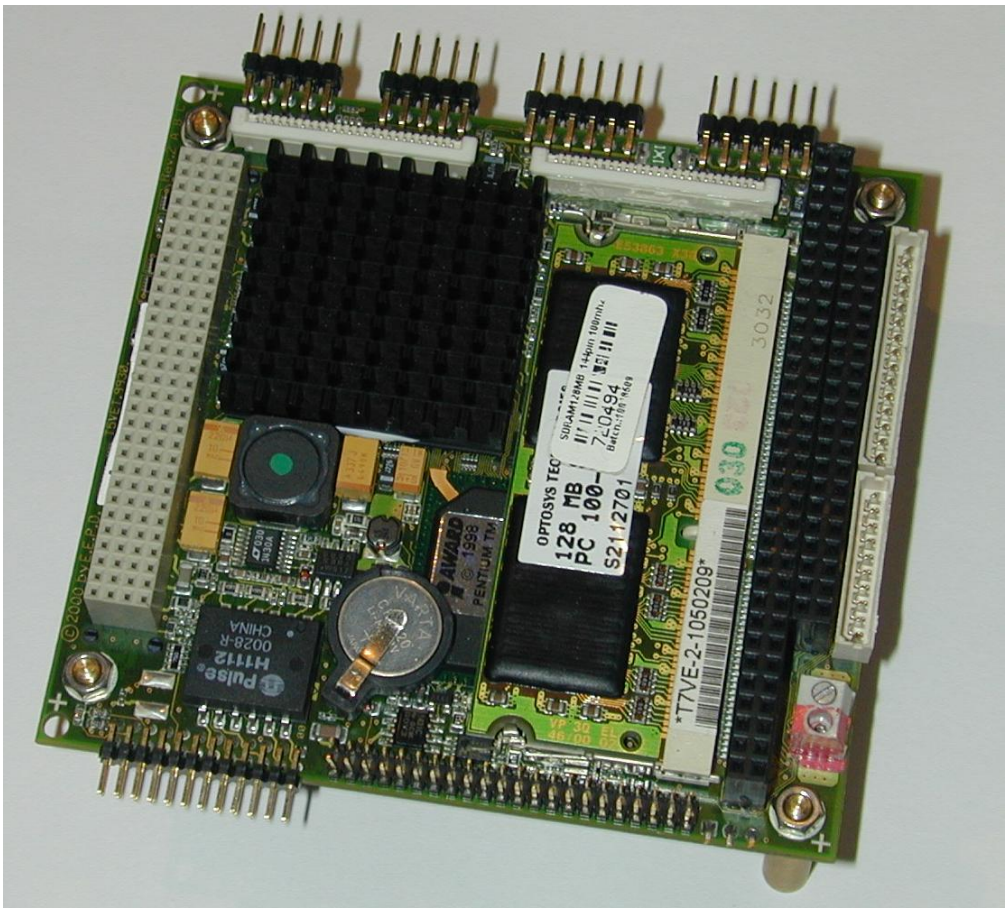


Figure 5: Industrial embedded PC based on Pentium MMX technology in PC104+ form factor

2.2 Operating System: Embedded NT

Embedded Windows NT 4.0 with service pack 5 was used to run the embedded PC. This operating system was chosen because it uses standard NT drivers and hence has wide range of driver support. Embedded NT also allowed us to select only the necessary parts of the operating system, compiling them and deploying the customized operating system into the embedded PC. This feature allowed us to exclude unnecessary OS components, thus minimizing the CPU operations and hence increased the system performance.

Additionally, Embedded NT allowed for self-logon during system boot-up while still providing good security screening for remote access requests. This allowed the system to boot up by itself during power up, to load and run

the necessary software including the graphical remote access server and to provide password authentication to access the system afterwards.

2.2.1 Compiling a Customized Embedded NT

Figure 6 shows the target designer for creating Embedded NT operating systems. Firstly, the developer selects and enables the desired operating system components at the “All Nodes” pane. These selections cover every aspect of the OS from general support such as hardware abstraction for different CPU platforms, support for different types of peripheral devices, and support for various management and networking to specific driver support of devices from third-party companies.

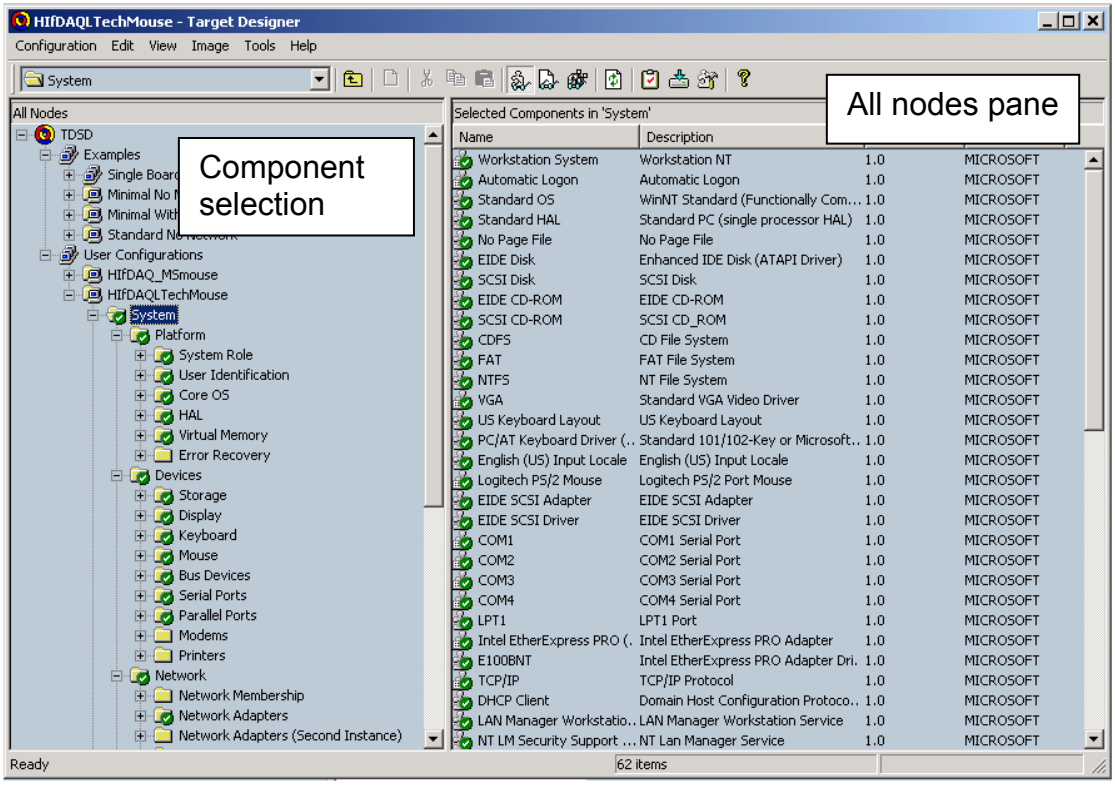


Figure 6: Embedded NT target designer software

After all the necessary components are selected and configured, the system is then compiled to generate an image of a working operating system ready to be deployed to the embedded system.

For the HiDAQ, the boot up memory was a small 2.5” harddisk containing the embedded NT operating system. The harddisk was prepared

by firstly formatting it with boot loader using a utility disk supplied with the Embedded NT installation package. After that, the entire operating system image was copied onto the harddisk. Alternatively, the harddisk can be first installed with a normal Windows NT operating system, which the embedded NT image will replace.

2.3 Remote Administration of the Data Acquisition

Three different control interfaces were provided by HiDAQ: the first was through direct user interface by connecting a monitor, a mouse and a keyboard to the system; the second was through graphical remote administration via an Ethernet connection; and the third was through prescheduled activities upon boot up for standalone operation. The first was useful for software development, debugging and testing, especially when working in the laboratory where the system was hooked up like a normal PC. For the direct connection, an underwater cable with appropriate connectors was designed to enable remote control at up to 10 meters away, which was especially useful for deployments near the sea surface or in test tanks. Operating in this mode provided a delay-free remote control; the only drawback was that the display quality dropped over range. This could be fixed by inserting a VGA signal booster circuitry in between, but is currently not implemented.

The second method was using Microsoft Netmeeting's Desktop Sharing via TCP/IP running over an Ethernet connection. With this, the users were able to logon into the Embedded NT, and to take control on its windows desktop, providing accesses to any applications within HiDAQ. A 50m underwater cable was designed for this purpose, allowing users to remotely perform data acquisition and download the data from HiDAQ to a remote system. The limitations of this method are the slow feedbacks from desktop graphic, keystroke, and mouse activity, which are not crucial for control purposes. This method was largely used in the experiment due to the combination of its flexibility and distance. When the system was deployed with this method, it was normally powered from the surface with 240V AC supply provided through the same 50-meter cable. A junction box with a built in AC

noise filter and on/off switch was provided at the surface end of the cable. Figure 7 shows a picture with the cable (1) to the HiDAQ casing, a standard cross-over signal RJ45 Ethernet connector (2) and a standard AC power cord (3).

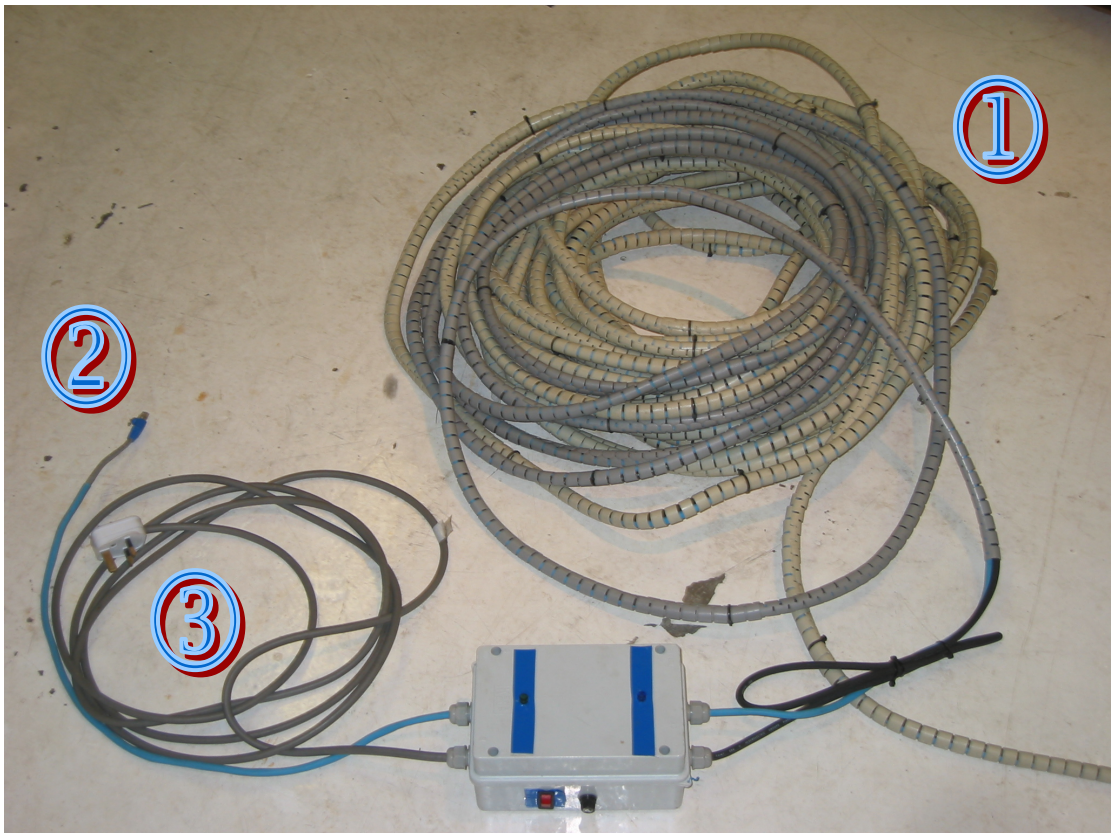


Figure 7: Fifty meter long underwater cable consisting a filtered power line and an Ethernet link

When HiDAQ is setup for standalone operation, a precompiled user acquisition program based on Labview®, a graphical programming language promoted by National Instruments, was loaded into the startup folder in the Embedded NT so that it would be automatically executed after it had been booted up. Acquisitions were performed based on the preset schedules in the program. The drawback of this method was that the users did not have access to HiDAQ during runtime. Nevertheless, this operation mode was necessary for standalone operation mode where no surface structure was nearby to support a user control station and no cabling was possible.

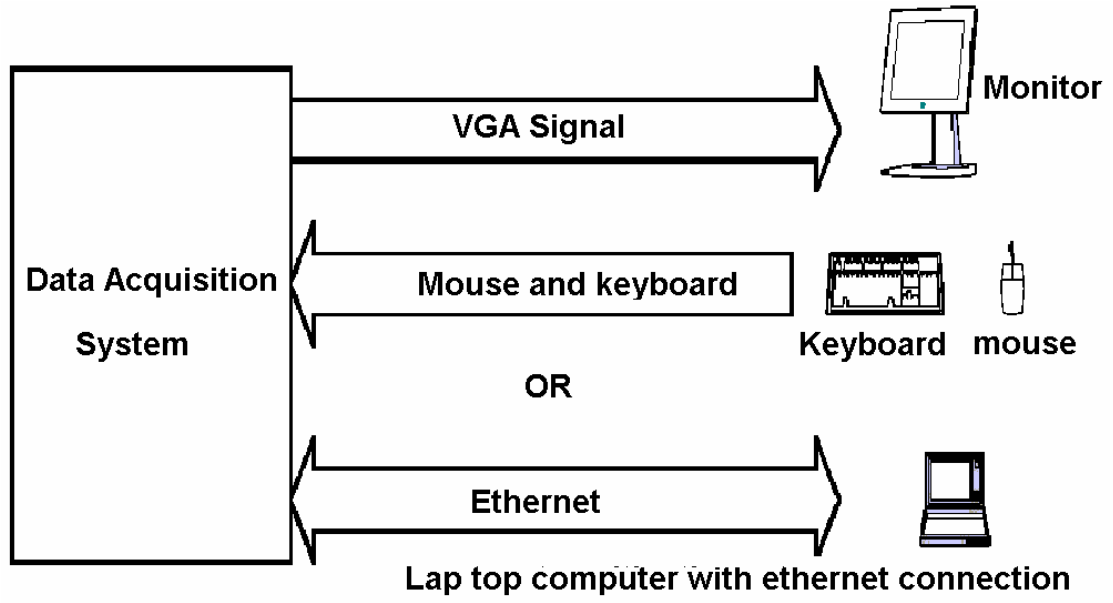


Figure 8: Different ways of controlling the HiDAQ

2.4 PCI Data Acquisition Card

HiDAQ used a multifunction I/O board from National Instrument (NI), part number PCI6110E, as it's analog digitization module. This card is a fully plug-and-play, full size PCI card for a desktop PC. It does not have DIP-switches or jumpers but is fully software configurable. It came with libraries of functions and APIs to control the acquisition card, including the board level hardware settings, both in Labview® and C language. The card could also be programmed using assembly language with the provided register level programming information.

The following settings could be configured through the software interface: sampling rate, input range, inter-channel sampling delay, and offset. The PCI6110E card is capable of sampling up to 5MSa/s aggregated and supports four simultaneous analog input channels. Nevertheless, the practical achievable throughput rate was limited by the overall performance of HiDAQ, which in turn was determined by the performance of each subsystem. The sampling rate was set to 500kSa/s per channel based on tradeoffs between getting a high sampling rate and the utilization of limited system resource such as percentage of system memory used as transfer buffer, sharing of CPU time with other supporting programs etc. This has provided enough

bandwidth to cover the frequency band of interest (up to 2MSa/s) and allowed continuous acquisition for reasonable time periods, while consuming less power and utilizing a smaller storage capacity.

The software also allowed the users to select different input voltage ranges in order to guarantee the usage of an optimal dynamic range. The acquisition card's Analog to Digital Converter's (ADC) input range is fixed at ±10V by the hardware; nevertheless the actual input voltage range could be adjusted by controlling the gain settings of the analog front end, see Table 2. Although the adjustable gain was able to scale the signal to ±50V, the maximum input rating of PCI6110E's analog front ends was limited to ±42V, in order to avoid saturation to the ADC; therefore the effective adjustable input range from ±200mV to ±42V. The inter-channel delays were set to zero in order to allow for synchronized recordings. All the input channels were set to AC coupling in order to remove any DC offset.

Table 2: Selections of input voltage ranges for analog input channels

Gain	Actual analog input range	Quantisation level
0.2	-50V to +50V (limited to ±42V by analog front end)	24.41mV
0.5	-20V to +20V	9.77mV
1.0	-10V to +10V	4.88mV
2.0	-5V to +5V	2.44mV
5.0	-2V to +2V	976.56uV
10.0	-1V to +1V	488.28uV
20.0	-0.5V to +0.5V	244.14uV
50.0	-0.2V to +0.2V	97.66uV

The PCI6110E card adds a wideband Gaussian noise to the input channels with r.m.s. amplitudes equivalent to half of an ADC bit to serve as a dither. Dithering causes the quantization noise to approximate a zero mean random variable rather than a deterministic function of input signal; as a result, the distortion of a small signal is reduced with the tradeoff of slightly increased noise floor [18]. This is particularly useful to detect the existence of

small signals with amplitudes within the order of the quantization level. This also significantly increases the analog channels' Signal to Noise Ratio (SNR) when recording stationary signals. This is achieved by averaging the acquired stationary signals, which will effectively increase the quantization resolution, improve the differential linearity and decrease the noise modulation. At ±5V input range, the noise level added is comparable only to the r.m.s. analog electronic noise (i.e. dither noise of 1.22mV compared to 1.7mV of the hydrophone signal conditioning analog circuit noise).

2.5 High-Speed Data Storage

A high performance SCSI160 PCI-SCSI host bus adapter (HBA) and an 80GByte SCSI160 harddisk were installed as the storage solution. The storage peripherals were selected to be a high-end system that requires minimum CPU intervention because the processor was chosen to run at relatively low clock rate (266MHz) to reduce power consumption.

A number of different solutions were investigated before this configuration was finalized, which included Fiber-Channel (FC), UW-SCSI and Fast EIDE. Recently, EIDE devices such as UltraATA100, UltraATA133 and serial IDE have been capable of high data transfer rates at 100Mbyte/s and above. Nevertheless, the IDE interface tends to occupy considerable amount of the system processing resources for a lot of its actions [19], which is therefore unsuitable for our configuration where limited CPU resources are available.

A 1Gbps (100Mbytes/s) fiber channel Storage-Area-Network (SAN) solution and a SCSI160 (160Mbytes/s SCSI-3) storage solution were tested. The performance of a FC solution (consisting of a QLA2200 host bus adapter and a Seagate 10,000rpm FC hard disk) and SCSI160 solution (consist of an Adaptec 19160 host adapter with Seagate 15,000rpm SCSI160 Hard disk) was compared using IOmeter: a vendor independent performance benchmark tool from Intel Corp. that is widely used in industry. The sustained throughput rate of the SCSI160 solution (achieving 40Mbytes/s) was found to out-perform the FC solution tested (25Mbytes/s), when only one harddisk was installed.

This comparison was not intended to benchmark the performance of both protocols but to find the best solution available during the time of system integration. This was because the performance is largely dependent on the combination of processor power, the specifications of harddisks used and the configuration of the harddisks. Based on the comparison, the SCSI160 solution was integrated into the HiDAQ, along with the OS and the user applications, to benchmark the overall performance. Different values of each acquisition parameters such as buffer size and data block size per harddisk write operation were adjusted and tested to find the optimum parameters for the best possible performance. The optimum parameters are shown in Table 3.

Table 3: Optimum acquisition parameters of current hardware configuration

Scan rate (for each channel):	500kHz
Buffer size:	35Mbytes
Number of scan per write operations:	800,000
Number of scan intended to acquire:	200M
Number of scan acquired before buffer overflow:	About 85M (170sec)

The system was capable of acquiring and streaming data continuously for a maximum of 170 seconds before the process had to be reinitialized. One of the reasons, apart from the limitation of low processor power, is the sharing of the PCI bus among the three devices (SCSI160 HBA, PCI6110E and Ethernet controller). To allow for longer acquisition durations, the acquisition software was written with a feature to recursively acquire bursts of data blocks with or without idle between the acquisitions. With this feature, we are able to perform data acquisition for durations that are as long as the capacity of data storage harddisk could support.

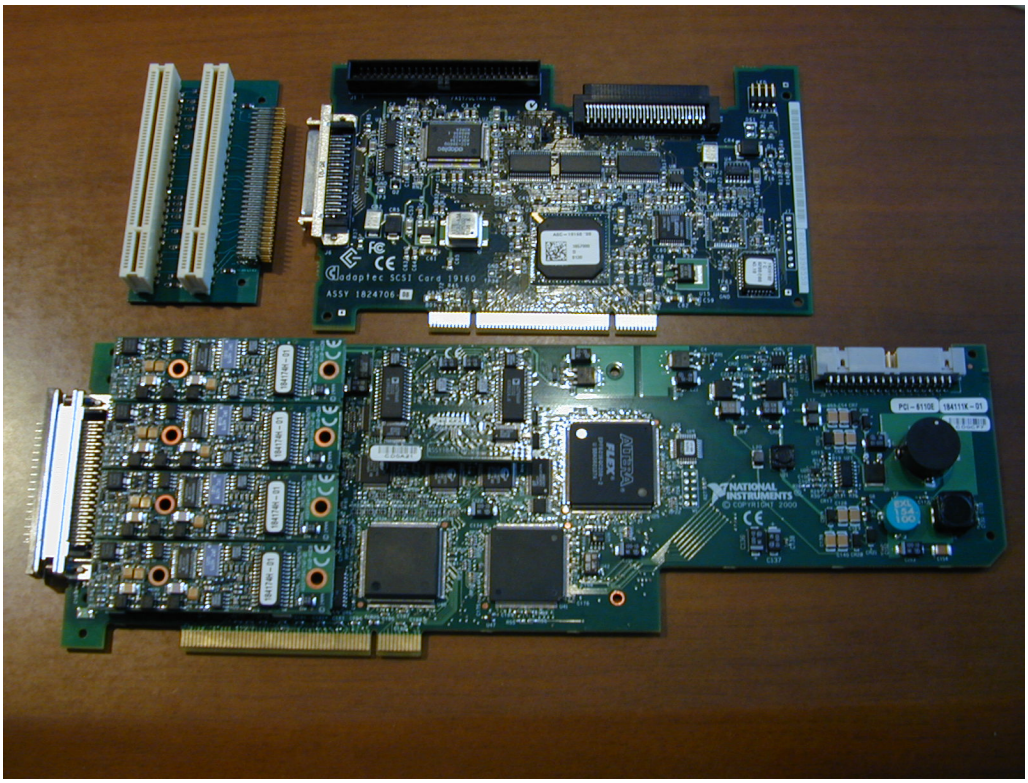


Figure 9: Data acquisition card, PC104+ to slot PC converter and SCSI160 host bus adapter

2.6 Power Supply Modules

The power supply module consisted of a low noise DC-to-DC voltage level converter and an energy source of either a high-density battery pack or AC-to-DC converter. The DC-to-DC converter was a high efficiency (up to 90%) PC104 module that provided a maximum combined power output of 90W and provided the desired voltage supplies of +5V (up to 10A), -12V (up to 0.5A) and +12V (up to 2A). Although the overall power consumption of HiDAQ was around 46W (see Table 4), a 90W DC-to-DC regulator was selected in order to provide a safety margin for the current surges during power up process.

Table 4: The maximum power consumption of the sub modules

Electronics Subsystem	Voltage (V)	Power (W)
N6110-PCI Data Acquisition Card	5	12.5
T-6VEF Pentium PC	5	10
2.5” System Hard Disk	5	2.5
SCSI160 Storage Hard Disk	5 12	4 9.6
PCI-SCSI160 Host Adapter	5	7.5
Analog board & misc.	±12V	0.3
Total Power Consumption		46.4

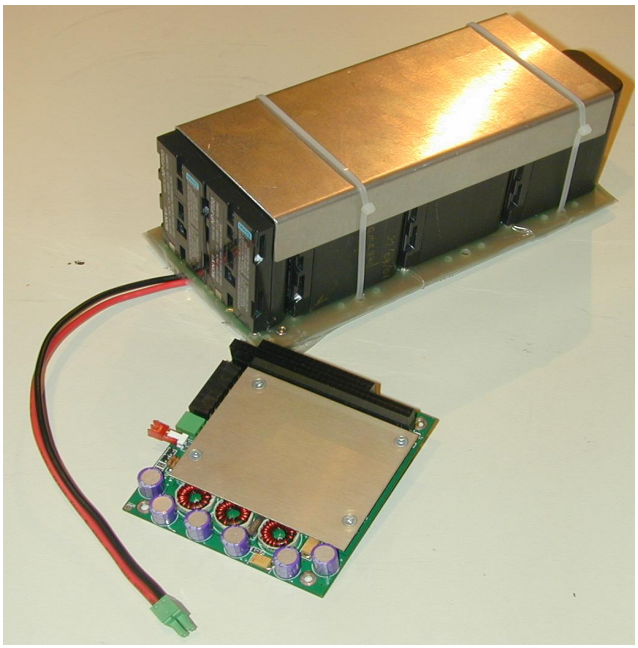


Figure 10: Power supply and battery

The power source could either be a battery pack or an AC-to-DC converter. The first option is suitable for standalone, short-term, operations while the later one is suitable for longer-term deployments at places with the existing of a nearby AC supply. The battery pack consists of six nos. 38.8Wh Sony infoLithium Lithium Ion batteries, providing 232Wh of energy to the digital circuitry, and two smaller Sony infoLithium batteries providing about 14Wh of energy for analog signal conditioning circuitries. As HiDAQ’s power consumption is 46.4W, the battery pack is capable of supporting the system

for up to 4.5 hours of continuous data recording. The maximum operating time could be extended if the recordings are temporally sparse. Figure 10 shows a picture of the battery pack and the PC104 DC-to-DC converter.

The second method of providing power to the system is from a 230V AC supply. This was implemented with a small size AC-to-DC converter in mini-ATX form factor. The DC-to-DC converter was inserted in between the mini-ATX module and the HiDAQ electronics. This ensured that the digital and analog power supplies were isolated in order to minimize the digital noise that was coupled to the analog board. Further more, an AC line filter was added before the mini-ATX module to remove any transients produced by the generator.

2.7 Analog Front-End for High Impedance Hydrophones

Although the PCI6110E data acquisition card provided it's own analog front end, it was not suitable for interfacing with high impedance sources like piezoelectric sensors. A high impedance source buffer was introduced between the hydrophone output and the analog front end of the acquisition card to minimize the impedance mismatch. Obtaining a clean signal from a high impedance source is rather difficult when the interested bandwidth is large and input signal level is very small. This is caused by the accumulation of the noises exhibited by all devices (such as op amp, filter etc.) across the signal conditioning circuits and since the signal level is small, these noises become significant.

The first stage op amp was selected with the lowest possible current noise, because when interfacing with a high impedance source, current noise becomes significant. Furthermore, any noise introduced near the sensor will go through the same order of amplification as the sensor signal and therefore should be suppressed efficiently in order to maintain high signal to noise ratio (SNR). The signal was then passed through filters and amplification circuitries before being feed into the NI6110's analog input.

2.7.1 Hydrophone and its High Impedance Piezoelectric Noise Model

The acoustic sensors used were reference class hydrophones model 10CT from GRAS Sound and Vibration. These hydrophones have an operation frequency range from 1 Hz to 170kHz and are reasonably omni-directional in all planes: horizontally (±2dB @ 100kHz) and vertically (±3dB@ 100kHz), except near the hydrophone housing (see Table 5). Unlike most other hydrophones that come with thick cables, the 10CT was supplied with a RF quality mini coax cable (about 2mm diameter). A 2mm cable diameter will minimize the scattering to any signal below 375kHz (signal with wavelength of twice or larger then the diameter and sound speed of 1500m/s). Nevertheless, its disadvantage is that it is relatively fragile due to the small size. In order to protect the cable with minimum disturbance to the acoustic sound field, it was put in a 10mm diameter silicone tube (see Figure 11). Silicone tubing was chosen because it is known to have acoustic impedance that is close to liquids and has been used in medical ultrasonic studies [20] and in underwater arrays [21]. The tube was then filled with caster oil that also has similar acoustic impedance as seawater; hence minimizing the distortion to the sound field.

Table 5: Simplified hydrophone specification

Receiving sensitivity (re 1uPa/V)	-211dB ±3dB
Frequency range	1Hz ~ 170kHz
Horizontal directivity	±2dB @ 100kHz
Vertical directivity	±3dB @ 100kHz (except near the cab housing)
Nominal capacitance	3.4nF
Max operating depth	700m
Weight	75g
Cable	6m with integrated LEMO SMB connector

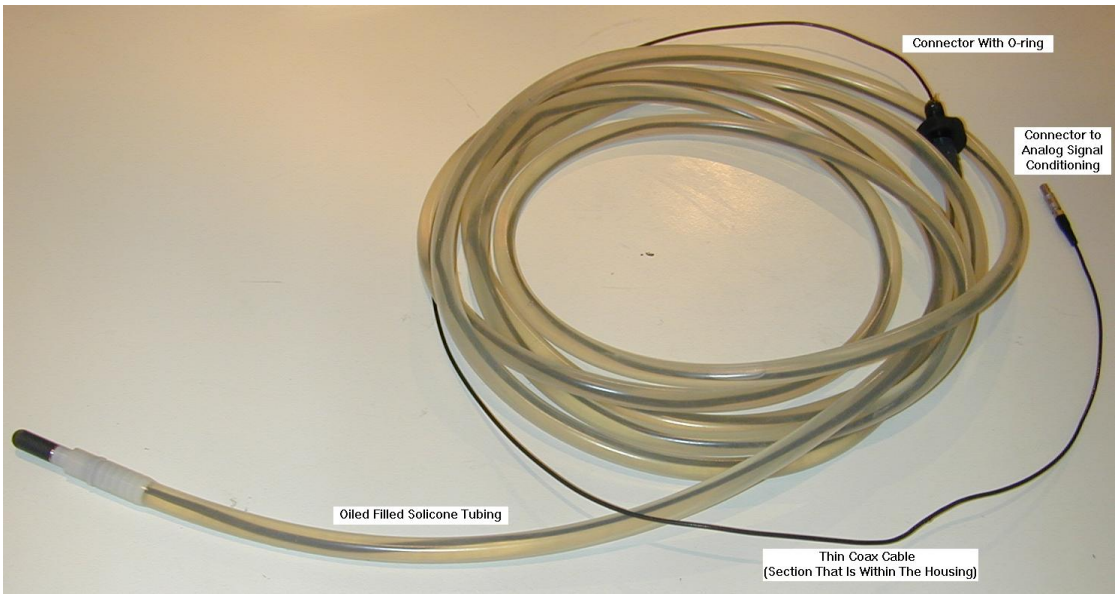


Figure 11: High performance hydrophone in protective cover

A piezoelectric sensor is generally characterized as a capacitor, which will generate charge when it is being stressed mechanically. An output voltage signal is generated when this small charge flows through an external high impedance load.

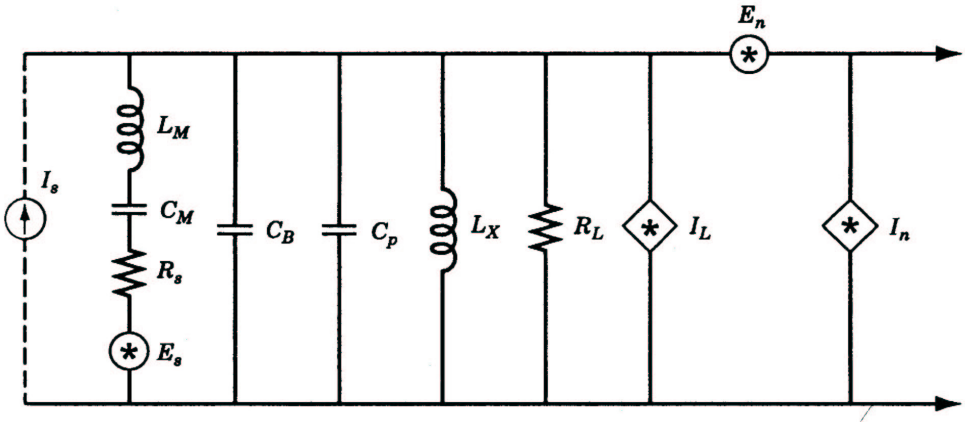


Figure 12: Noise equivalent circuit of a Piezoelectric Sensor (from low-noise electronic system design by Motchenbacher & Connelly [22])

Figure 12 illustrates an equivalent schematic of the noise model of a piezoelectric sensor. L_M is the mechanical inductance; C_M, mechanical capacitance; and Rs is the series loss resistance. These three terms model the generation of electrical output by changing the reactance of the system with respect to mechanical stress. L_X is the external inductance; C_B is block or

bulk capacitance; C_P is the cable capacitance; R_L is the load resistance; and I_S is the current source of the signal. The noise parameters of operational amplifier and its network around it are represented by E_n (its voltage noise) and I_n (its current noise). The equivalence input noise of this transducer can be represented by Equation 1, which is simplified and adapted from [22].

$$E_{ni}^2 = E_S + E_n^2\left(\frac{Z_S + Z_l}{Z_L}\right)^2 + \left(I_n^2 + I_L^2\right)Z_P^2,$$

Equation 1

Where,

- Zs is the series impedance of hydrophone: R_S, L_M and C_M
- Z_L is the parallel impedance of C_B, C_P, L_X, and R_L
- Z_P is Z_L in parallel with Z_S.
- E_S is the thermal noise of R_S (given by $4kTR_S$)
- I_L is the thermal noise of R_L (given by $4kT/R_L$)
- E_n, I_n are the voltage and current noises

The E_S term is neglected because R_S is small. This leaves the voltage noise E_n, and the current noise, I_n. Again, the E_n contribution is normally small with respected to noise generated from I_n for high impedance devices [23] [24]. Since the total noise power is the sum of square of all uncorrelated noise sources, any source that generates more than 5 times the noise of other sources will dominate, which means in this case, I_n will dominate.

Z_P is large at relatively low frequencies (caused by the impedance of C_B and C_P). In order to minimize the current noise contribution, R_L should be kept large and I_n small. Since current noise can be termed as $\sqrt{2qI_B}$ A/√Hz [27], where q is the electronic charge, op amps with small bias current (the I_B term) such as BiPolar devices (with minimum collector current) or FET devices (with minimum leakage current) are good candidates The I_nZ_P term is normally prominent at lower frequencies and will be insignificant at higher frequency as I_n is a 1/f noise. Here, FET is a better choice since it normally has less of a low frequency 1/f component in its current noise. Furthermore, FET normally requires less or no biasing and so R_L can be high which matched our requirements.

2.7.2 High Impedance Analog Front-End

Several potential op amps such as LT1169, AD743, LT1793, LT1113 and INA116 were identified and evaluated to decide which op-amp would be most suitable in terms of noise and gain-bandwidth product.

Based on the performance along with other considerations such as small packaging size, implementation limitations etc., LT1169, a JFET op amp was selected for the analog front end. It was chosen because of its low voltage noise (that is comparable to the performance of a bipolar device) while maintaining the low current noise of a FET device at the same time, which were 6nV/√Hz and 1fA/√Hz respectively. Apart from its optimum noise performance, it has a very high input resistance of $10^{13}\Omega$, a low input capacitance of 1.5pF and a large Gain Bandwidth Product of 5.3MHz. The output offset was relatively high (2mV) for a first stage solution but this was rectified by offset nulling, employing a DC servo circuitry.

There are two main categories of high impedance transducers: capacitive transducers and charge emitting transducers. Hydrophones fall into the later category. There are two main approaches to translate the input charge variation of and charge-emitting device to an output voltage change: the first is through a charge amplifier and the second is by using a high impedance voltage follower. The high input impedance follower has the advantage that its noise gain can be controlled easily, thus achieving better noise performance. The disadvantage is that it is sensitive to any intermittent capacitance between the piezoelectric and its input, limiting its applications to scenarios where relatively short cables are used between the transducer and first stage electronics. In contrast, a charge amplifier is insensitive to intermittent capacitance; hence it is suitable for applications where there is long cable between the high impedance transducer and the front-end analog circuitry. Nevertheless, the disadvantage is that its circuitry noise is generally higher than the high impedance voltage follower. As the cables between the hydrophones and the analog circuit board were relatively short, the high impedance follower circuit realization (which is basically a virtual charge

amplifier) was implemented to take advantage of its lower noise characteristics.

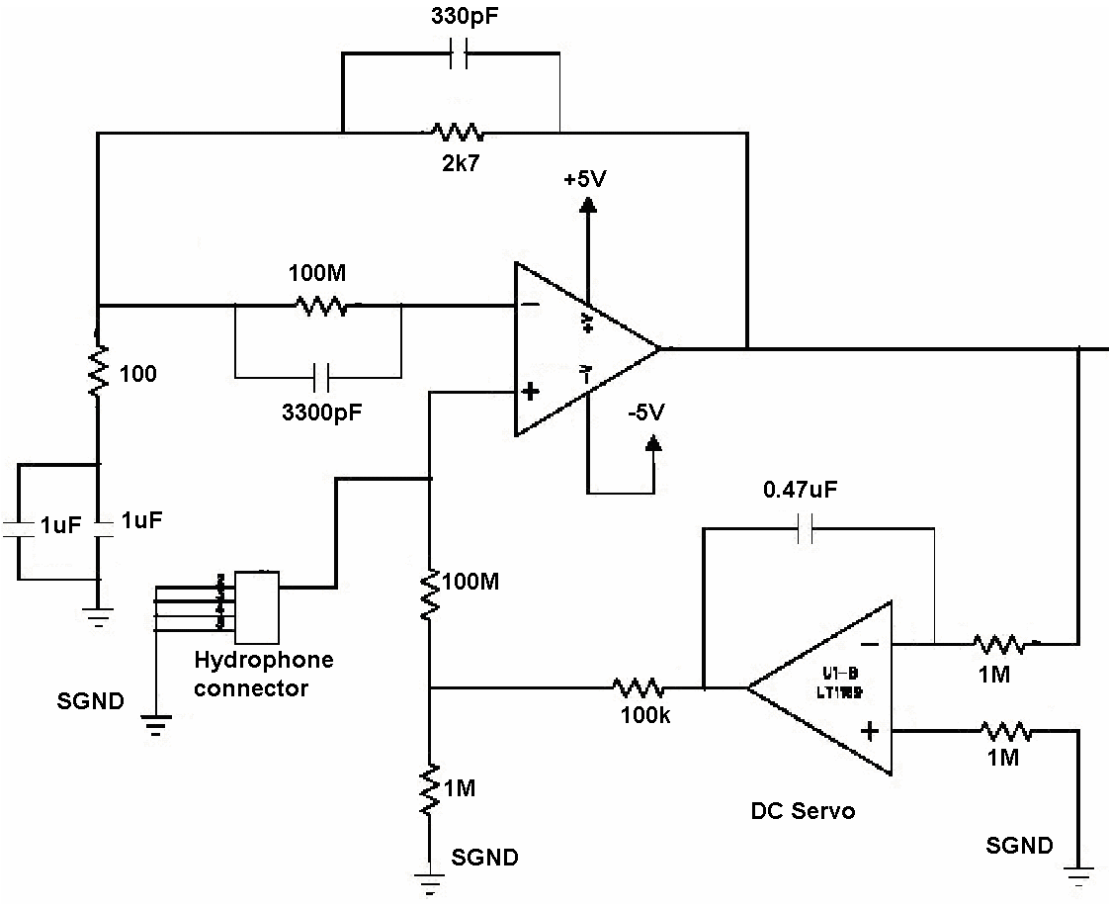


Figure 13: High Impedance Voltage Follower with DC Servo and Integrated High Pass Filter

Figure 13 shows the schematic of the analog front end in voltage mode. This first stage provided an overall gain of about 28dB to the sensor signal. The current noise of the internal bias circuitry in the op amp could get coupled into the input signal via the FET's gate-to-source capacitance and would then appear as extra input voltage noise. In order to cancel it, a similar bias current at the other input was needed. Therefore, an equivalent capacitance that matched the sensor's capacitance was introduced to the op amp's inverting input to provide a compensation effect.

A drawback of the LT1169 was that it presented a relatively high dc offset (up to 2mV); this was unacceptable for the first stage circuitry, as it would have reduced the effective dynamic range. A DC servo was

implemented to rectify this issue, making sure that any dc offset would be compensated so that the output voltage would always swing around the signal ground.

The power supply for the LT1169 was kept at ±5V although the maximum rating of this device was ±20V. This was done so that the gate-to-junction leakage current was reduced and the heat generation was minimized at the same time. Precautions were also taken to filter the power supply with a simple LC network in order to remove noise and harmonics.

2.7.3 Analog Stage with Pre-selectable Gain

The analog output of the first stage, the high input impedance voltage follower, was a low impedance signal. This meant that the noise characteristic requirements of subsequent op-amps stages had changed from a low current noise to low voltage noise. The reason was that when interfacing to a very low impedance source (output impedance of the LT1169), the voltage noise contribution is dominant and contributions from other sources can be neglected [25]. Therefore, an AD797 was selected to implement the gain stage. The AD797 provided ultra low harmonic distortion (-120dB at 20kHz), very low voltage noise (0.9nV/√Hz), and a high gain bandwidth.

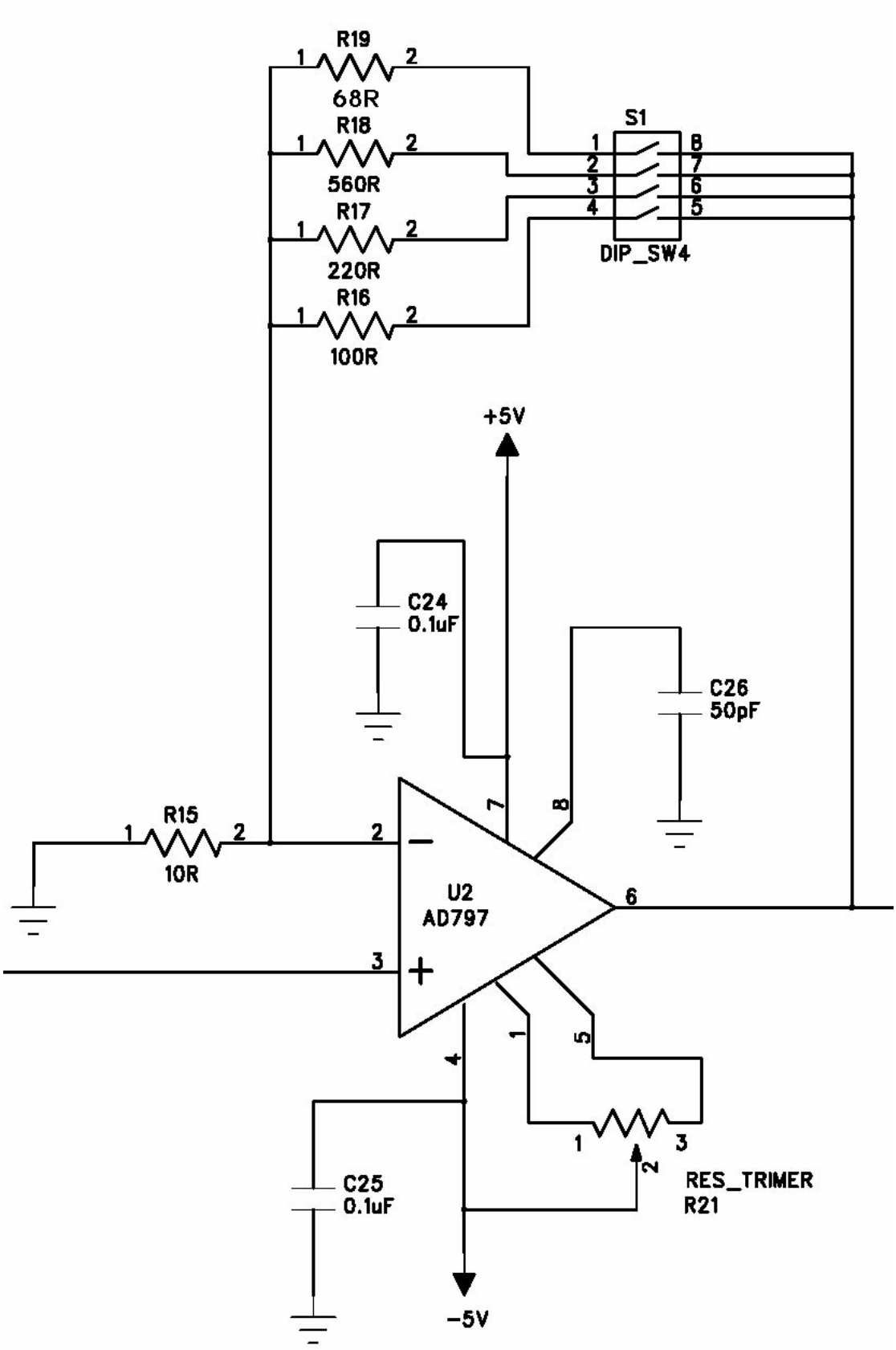


Figure 14: Low Noise Selectable Gain Stage

The high gain bandwidth product of AD 797 enabled us to use a single device to implement the gain stage and therefore optimize the noise performance. This made sure that the minimum number of components was used and reduced the number of routing traces was needed during PCB routing. Hence, the number of electronic noise sources was reduced and the possibility of interference noise was minimized. Although the gain bandwidth product varied at different gain value and compensations [26], it was possible to provide at least 300 times gain at 300kHz. The AD797 was capable of operating stably with external resistor networks that were very small in value. This effectively improved the overall noise performance by significantly reduced the total noise caused by current noise and thermal noise at op-amp's external network. Besides providing offset compensation pins, the AD797 also provided access to its internal compensation network, which could be modified by adding external capacitors. This effectively improved the distortion performance and the gain bandwidth by providing appropriate compensation. The schematic is shown in Figure 14. The gain stage provided a DIP-switch that allowed the user to manually select different amplifications; choices available were 10x, 22x, 56x and 6.9x.

2.7.4 High Pass and Anti-aliasing Filter

A band pass circuitry was implemented to remove low frequency signals below 1kHz as well as any signal above 250kHz. The first was achieved with a simple 2-pole active high pass filter. This made sure that low frequency signals (which are dominated by shipping noise) [1] did not saturate the dynamic range. Figure 15 shows the schematic of the high pass filter. Similar routing techniques and power supply filtering measurements used in the gain stage was duplicated in this circuit.

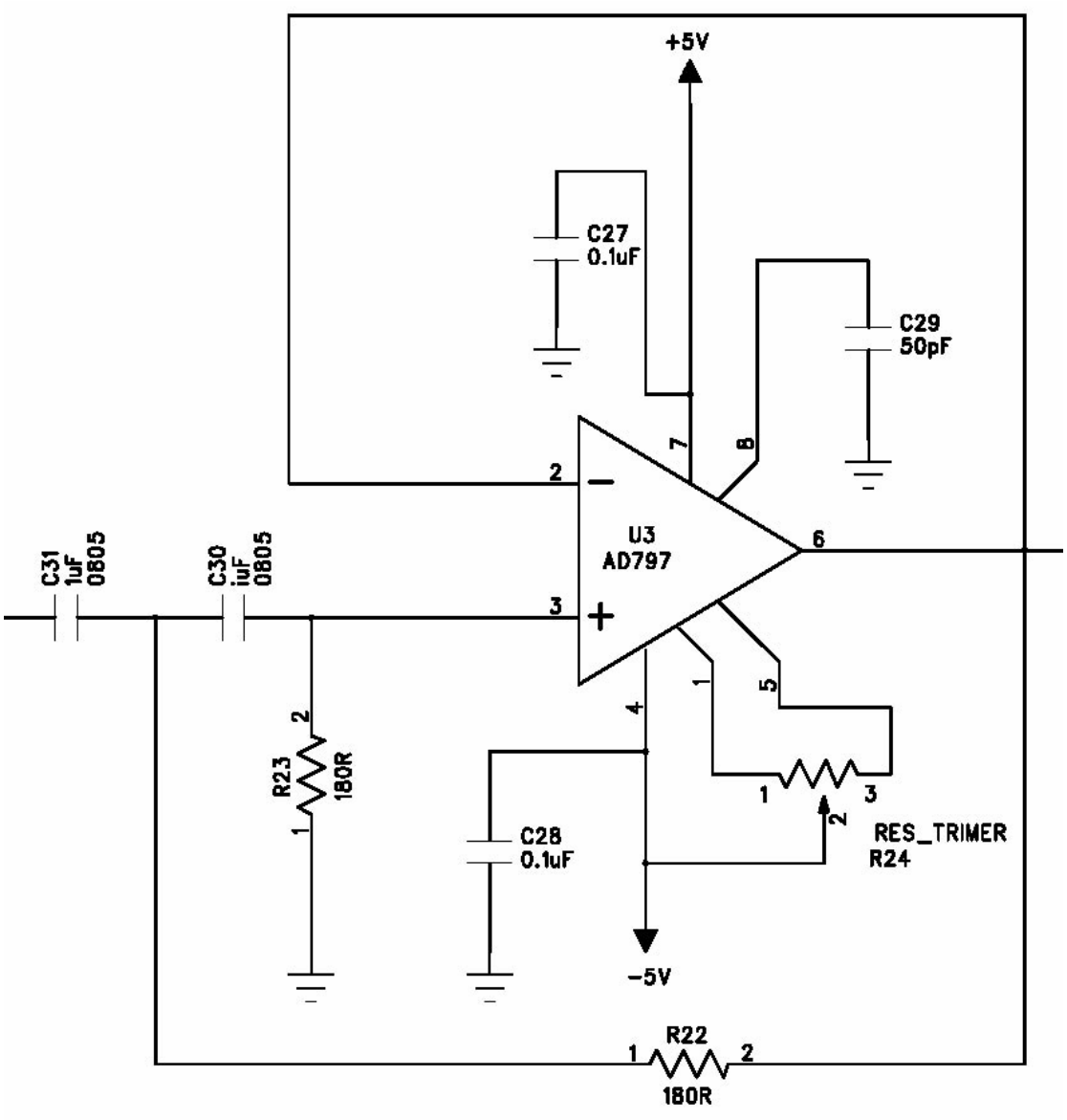


Figure 15: Schematic of High Pass Filter

The signal was then passed through a sharp low pass filter before being fed into PCI6110E PCI card to avoid aliasing. To prevent signal distortion, a low pass filter with linear phase response was desired. Since linear phase filters normally have shallow initial attenuation curves and hence are relatively inefficient in providing the required steepness beyond the cut-off frequency, a higher order filter was chosen.

An 8^{th} order low pass filter was implemented using a monolithic RC continuous filter IC (Figure 16). Although the hydrophone response is specified up to 170kHz, the filter's 3dB cut off frequency was set to 200kHz. This ensured that the frequency region with a long group delay would fall

outside the frequency band of hydrophone response and created a less than 5μsec group delay from DC up to 170kHz (see simulated low pass filter response for details, Figure 17**)**. The 8^{th} order low pass filter provided 64dB stop band attenuation at 250kHz, making sure no appreciable energy was remained above 250kHz. Since the acquisition card was sampling at 500kSa/s, aliasing was negligible while the phase of signals of up to 170kHz was kept linear. Although there was signal between 170kHz and 200kHz, the response was not specified by the hydrophone, and the group delays were high. The signals within this frequency range could be used, but the hydrophone response should be calibrated and the group delay of the filter circuit should be compensated.

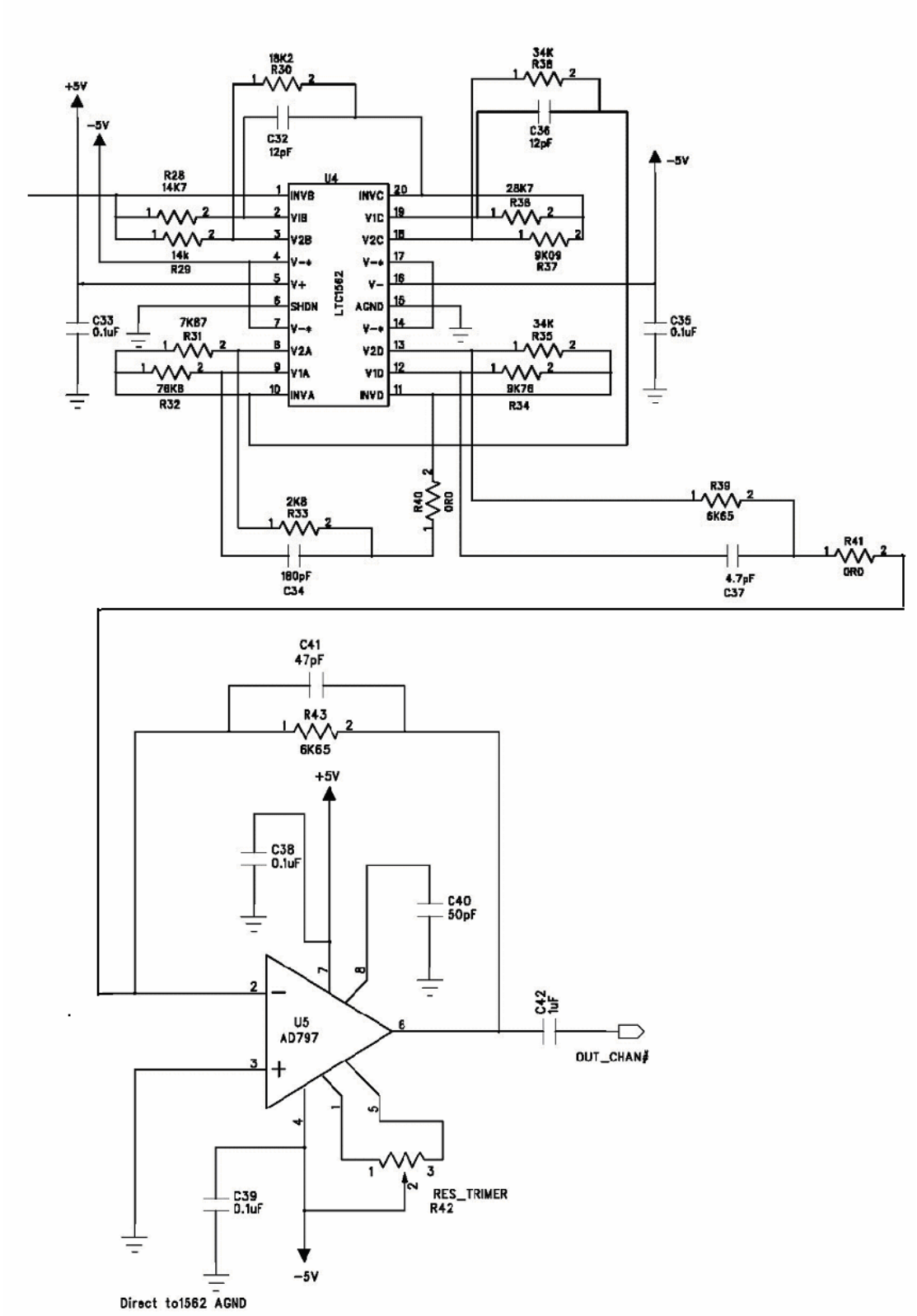


Figure 16: 8th Order Low Pass Filter (LPF)

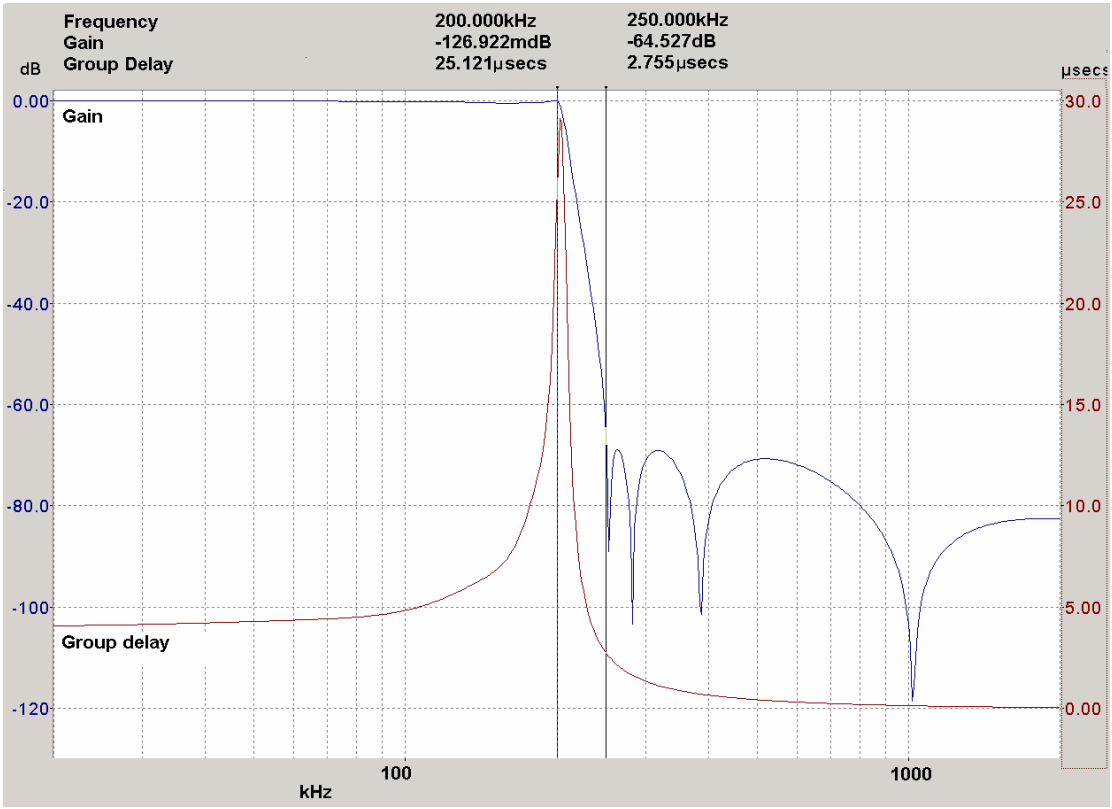


Figure 17: Frequency Response and Group Delay for the low pass filter

2.7.5 Printed Circuit Board Design

The analog board was routed in such a way that the current return paths of large or noisier signals (such as the outputs of the op amps and monolithic filter) did not interfere with the small signal current return paths (such as the signal of the op amp input pins). The larger signal circuits were placed closer to the power supply (refer Figure 18) so that its return current (see the thick orange loop) did not disturb the return current of small signal (the thin green loop). Furthermore, low impedance paths between the power plane and ground plane were provided by adding bypass capacitors between them near the power supply input of the active components of each of the stages. A ground plane was inserted between every pair of signal layers or power layers and a sufficient number of via holes were provided at strategic places between the ground planes to provide solid grounds.

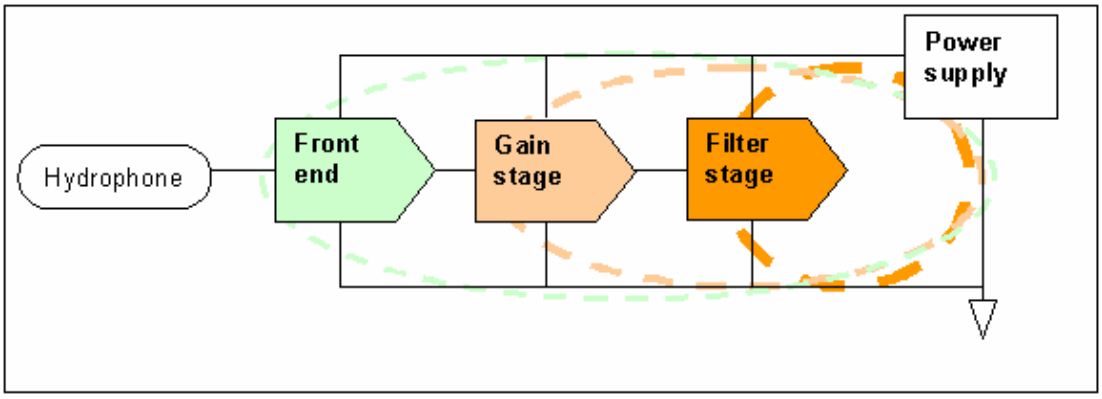


Figure 18: Current return path of different of various analog stages

Guard-rings were provided to traces that directly connected to the hydrophone output, minimizing electromagnetic disturbance to these very small signals. Since the input signals of the high impedance voltage followers (the first stage) were sensitive to the intermittent capacitance, solder masks at the input area were removed.

The filter stage electronics were placed after the gain stage so that the noise from networks in filter stage did not get amplified. All circuits of the four channels, signals, power and ground planes, were isolated from each other while the only connecting points between them was the inlet of the power supply (see Figure 19), which is heavily filtered by capacitor networks. This was done to minimize the noise coupling and the cross talk between channels. Lastly, all the components used in this board, except for the voltage regulators and connectors, were surface mount components so that a good noise performance and compact board size could be achieved at the same time. The use of thru-hole components was avoided because they would introduce inductance at the leads, which could increase the system noise.

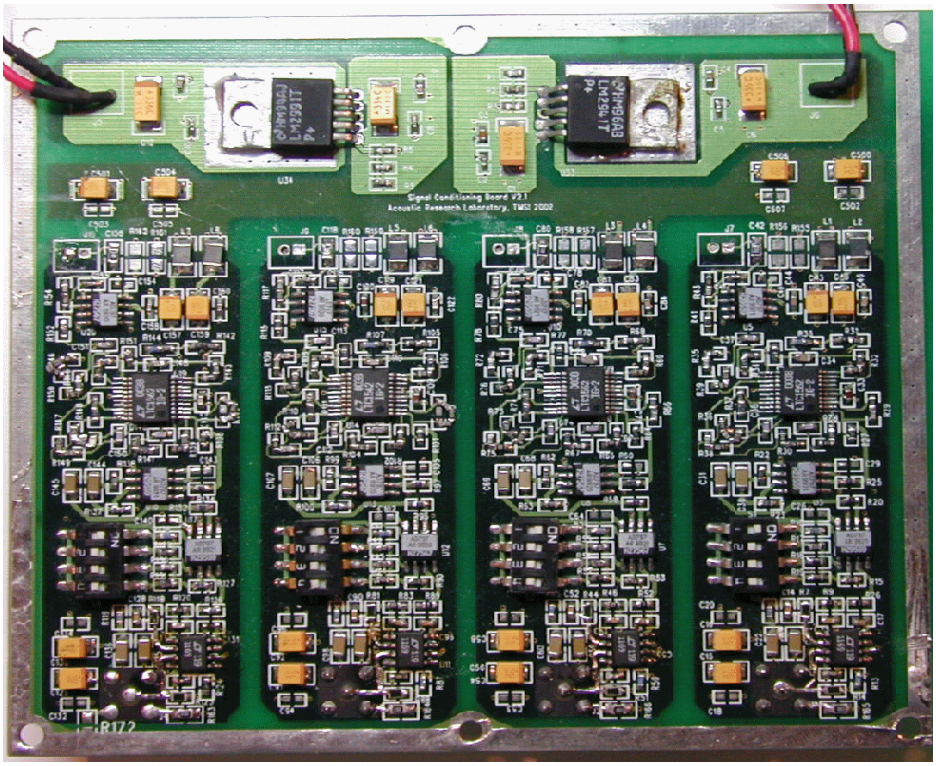


Figure 19: PCB for the analog signal conditioning

2.8 Prototype Electronics Performance

The following paragraphs discuss the ideal performance derived from theoretical calculations given the specification of components used. These ideal values were then used as specification targets when designing the analog signal conditioning board. A comparison between these ideal values and the performance that was practically achieved is given in this section.

2.8.1 The Noise of the Analog Board

According to specs, the LT1169 produces a voltage noise of about 6nV/√Hz @ 1kHz when loaded with piezoelectric sensors of capacitances between 100pF and 5000pF; this applies to our case as the 10CT has an equivalent capacitance of 3400pF. Although the LT1169 voltage noise curve tends to have higher noise levels at lower frequencies, it tails off with the 1/f shoulder around 100Hz and remains less than 7nV/√Hz onwards. Since the analog board provided a two-pole high pass cut off around 1kHz, providing good enough attenuation at low frequency to remove the signal with high noise, therefore 6nV/√Hz is a good approximation. By assuming the passive

networks were consist of only resistances, and LT1169 had 1fA/√Hz current noise, the equivalent r.m.s. noise (in voltage) generated by the first stage preamplifier due the thermal noise, voltage noise and current noise of the entire network (including the hydrophone) is about 56μV/√Hz (as a comparison, it would be 200μV/√Hz if we use AD797), obtained using Equation 2.

$$V_{noiseFirstStage} = A_{VfirstStage}\sqrt{V_{nosieLT1169}^2 + 4kTR + (I_{noiseLT1169}R)^2}$$

Equation 2

Where, $A_{VfirstStage}$ is the voltage gain of the first stage, 28x
$V_{nosieLT1169}$ is the voltage noise of LT1169 Op Amp, 6nV/√Hz typical
$4kTR$ is the Johnson noise,
k is the Boltzmann's constant,
T is the operating temperature, assuming 313°K (50°C)
R is the equivalent input resistance networks, about 100MΩ
$I_{nosieLT1169}$ is the current noise of LT1169 Op Amp, 1fA/√Hz typical

56μV/√Hz is a large noise to exist in the first stage and must be significantly reduced in order to measure the ambient noise. The noise level was significantly reduced by adding capacitor networks to the op-amp circuitry as suggested by the specification sheet of LT1169 (which stated that the noise could be brought down to 128nV/√Hz at 20x gain with circuits having equivalent resistance of 100MΩ or more) [27]. Our first stage had a gain of 28 times; therefore, the first stage was assumed to generate an overall total noise of approximately 179nV/√Hz (300 times smaller).

At the gain stage, the accumulated r.m.s. noise was calculated as [26]

$$V_{noiseSecondStage} = \sqrt{V_{noiseAD797}^2 + 4kTR_s + 4\left(I_{noiseAD797}R_s\right)^2 + V_{noiseFirstStage}^2 \times A_{V\,\mathrm{sec}ondStage}},$$

Equation 3

Where, $A_{VsecondStage}$ is the voltage gain of the second stage (the gain)
$V_{nosieAD797}$ is the voltage noise of AD797 Op Amp, 6nV/√Hz typical
$4kTR$ is the Johnson noise,
T is the operating temperature, assuming 313°K (50°C)
Rs is the equivalent input resistance networks, 10Ω
$I_{nosieAD797}$ is the current noise of AD797 Op Amp, 2pA/√Hz typical
$V_{noiseFirstStage}$ is the total noise from the first stage, about 179nV/√Hz

Ignoring the Johnson noise and current noise of the AD797 due to the small resistor values, and with a gain of 10 times, we yield a total noise of about 1.8uV/√Hz. As the signal was fed through the bandpass network consisting of a 2^{nd} order high pass with 3dB cut off at around 1kHz and an 8^{th} order low pass with 3dB cut off at around 200kHz, we approximated the bandwidth to be 210kHz considering the skirts of the attenuations at both sides, and the noise was calculated to be around 0.82mV. Noise levels generated from filtering circuits were very small (about 39μVrms over bandwidth of 400kHz [28]) and were ignored in this theoretical estimation. This noise level estimation was also derived by assuming ideal circuit construction, without taking into account any noises introduced by soldering, interference picked up by the traces, the noise introduced by cable interconnections and noise from power supplies etc.

The total noise level of the entire customized front-end analog and signal conditioning circuit was measured to be 1.3mVrms~1.5mVrms (11mV~15mV peak to peak) at a gain of 280x (28x10), which consumed the last 3 bits to toggle at the peak to peak noise, but toggled less than 1 bit at r.m.s. value (the ADC resolution was 12bit and input voltage range was assumed to be ±5V). The empirically measured r.m.s. noise value was almost twice the ideal noise performance calculated and was considered acceptable. Therefore the gain setting of 280x was optimized when used with 12-bit data acquisition system set at 10V peak-to-peak input range because the noise level occupies only ½ LSB (Least Significant Bit) of the system.

Nevertheless, the gain setting used in field trips could sometimes be higher than this so that ambient noise peak level will fill up at least 50% of the dynamic range most of the time. Although the analog noise floor will be raised respectively, we would still benefit from the signal processing gain if we are able to extract the transients within the noise floor.

2.8.2 Analog Channel Transfer Function

The transfer function of the HiDAQ analog board has been empirically measured using a SR785 network analyzer. This allows us to correct the

signal below 100kHz to for the transfer functions of the electronics. Although the operating frequency of the board was up to 200kHz, the transfer function of the analog was measured only up to 100kHz due to the bandwidth limitation of the network analyzer.

Figure 20 shows the typical frequency response of the analog signal conditioning board. The curve shows that there were 2 high pass cut off frequencies: one around 800Hz, another one around 1.2kHz. The first was the high pass produced by the first stage, and the later was produced by the active high-pass circuitry. The frequency response has a slightly negative slope, with 3dB signal loss from about 1.2kHz to 100kHz, which could be compensated digitally when the data was being analyzed.

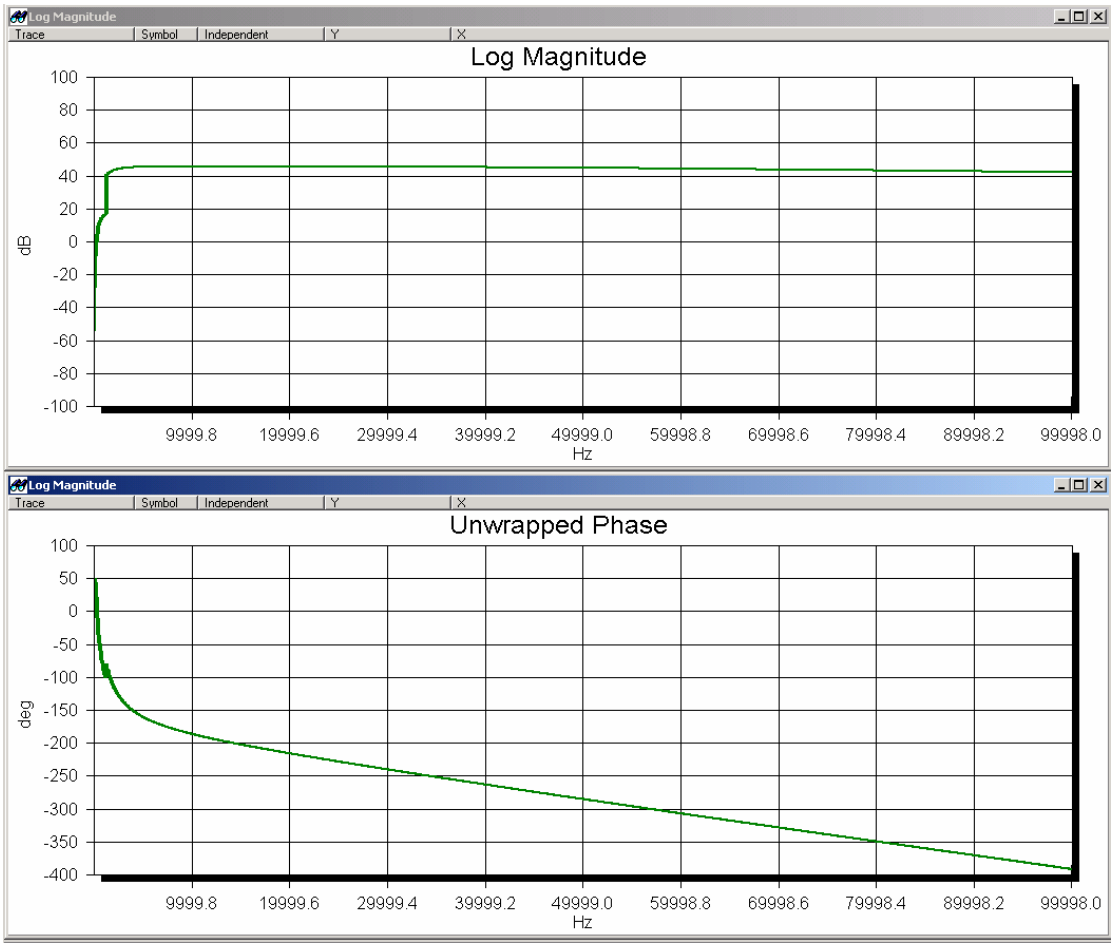


Figure 20: Typical frequency response curve of the analog board

Table 6: Measured gain of each channel at different setting

Channel	Gain SW1 (dB)	Gain SW2 (dB)	Gain SW3 (dB)	Gain SW4 (dB)
Desired gain	48.9	55.8	63.9	45.6
Ch0	49.4408	55.7614	63.7590	45.7740
Ch1	49.2711	55.5908	63.4937	45.5810
Ch2	45.7740	55.8134	63.6772	45.8323
Ch3	49.4986	55.8496	63.7228	45.8096

The maximum gain of the individual channels of the analog board (typically near 1.2kHz) is presented in Table 6. All the gains achieved from the circuitry were within 1dB accuracy from intended value with 0.5dB tolerance except Channel 2, when set at 48.9dB gain, gave a difference of 4dB. This is due to the components errors in the gain stage.

2.9 Heading and Pan-and-tilt Sensor

As the system could be deployed at any directions and tilt angles, it is crucial to know its three dimensional orientation when mapping the sources. An OEM electronics compass and 2 axis-level sensors were integrated onto the frame of the array in order to provide heading and orientation information to the array. Nevertheless, a prototype electronic compass with the same sensors has been built and tested at the beginning stage of this project the analysis is presented here.

2.9.1 Basics of a Tilt Compensated Electronic Compass

The strength of the Earth's magnetic field is about 0.5 to 0.6 Gauss in open air pointing towards the Earth's magnetic North pole from the Magnetic South. Therefore, an array of magnetic sensors sensitive to 70μGauss or better should be able to achieve an accuracy of 0.01° (derived from the inverse tangents of 70μGauss/300mGauss) at a horizontal plane near the equator. The magnetic field also has dip angles; where the magnetic field lines are not parallel to the earth's surface anymore (pointing up at Southern

Hemisphere, and pointing down at Northern Hemisphere), see Figure 21. These effects are minimal within Singapore region as it is near the equator, but for areas that are away from equator, it is important to know their geographical locations and compensate for this error. For a tilt compensated compass, the Earth's three-dimensional magnetic flux (horizontal and vertical) and the compass's gravitational pitch and roll orientations must be measured. This was done with three perpendicular magnetic sensors and a dual axis level sensor.

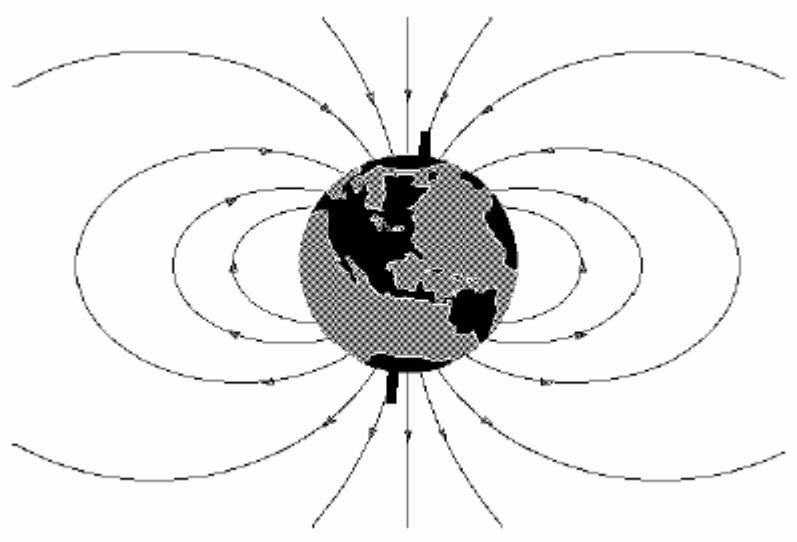


Figure 21: Magnetic field of the Earth (adapted from application note by Caruso, Honneywell)

The sensors used in this prototype were Honeywell's HMC1001 (single axis) and HMC1002 (dual axis) magneto resistive sensors, which have a resolution of 40μGauss. Although the resolution of the sensors were good enough to generate very fine heading resolution, the actual heading accuracy achieved would depended on how the magnetic field distortion (due to nearby hard/soft iron) could be compensated, the extend of the compass tilting, the declination angles and the noise of analog circuits.

When leveled, the horizontal component of the Earth's magnetic field is parallel with the sensors' X-Y plane. Therefore, the values measured by the two axis sensors (Hx, and Hy) could be modeled by the $\cos(\theta_{heading})$ and $\sin(\theta_{heading})$ functions where $\theta_{heading}$ is the heading referred to the magnetic North (see Figure 22). Therefore, relative compass heading could be obtained

by calculating the arc tangent of the ratio Hx/Hy, assuming that there were no nearby ferrous materials around. In this case, the compass heading could be determined with the following set of equations in Table 7 [29]:

Table 7: Compass heading calculations

Compass heading in degree	Condition
90	Hx=0, Hy <0
270	Hx=0, Hy >0
180-[arcTan(y/x)]*180/π	Hx <0
-[arcTan(y/x)]*180/π	Hx >0, Hy <0
360-[arcTan(y/x)]*180/π	Hx >0, Hy >0

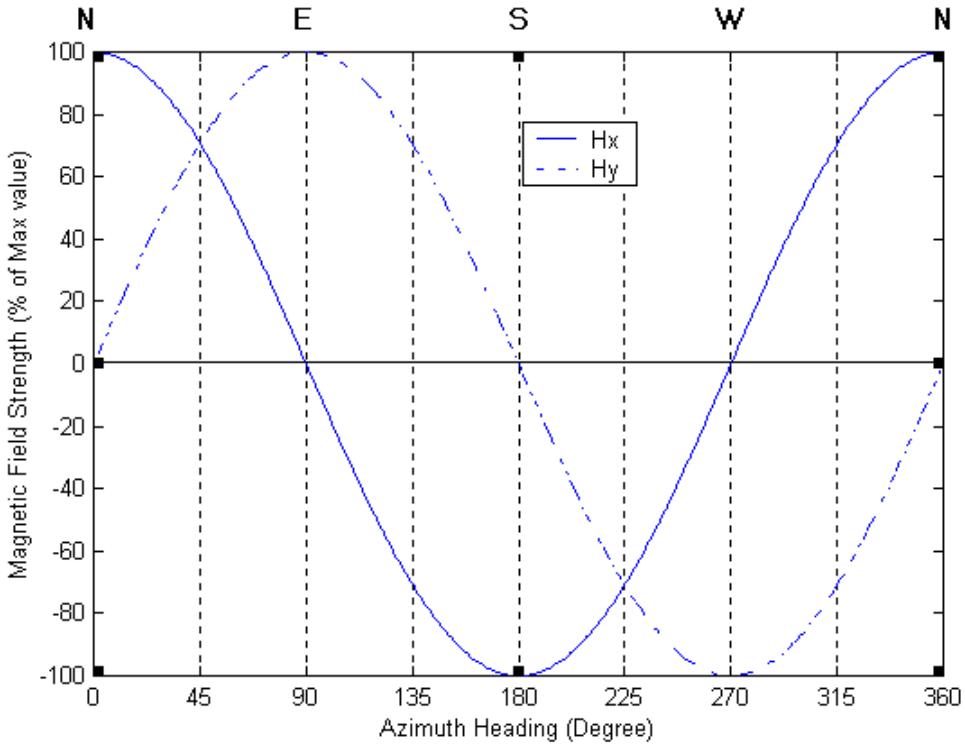


Figure 22: Ideal X-Y reading of the Earth's horizontal magnetic field

When the compass is not gravitationally leveled (see Figure 23), the magnetic field values measured are deviated as the sensors are measuring the Earth's horizontal magnetic field from an angle. To compensate for these deviations, a third magnetic field component orthogonal to the compass (Hz)

is needed, along with the pitch ($\phi_{compass}$) and roll ($\theta_{compass}$) angle of the compass. The azimuth magnetic components are then recomputed using Equation 4 and Equation 5 and the headings are recalculated.

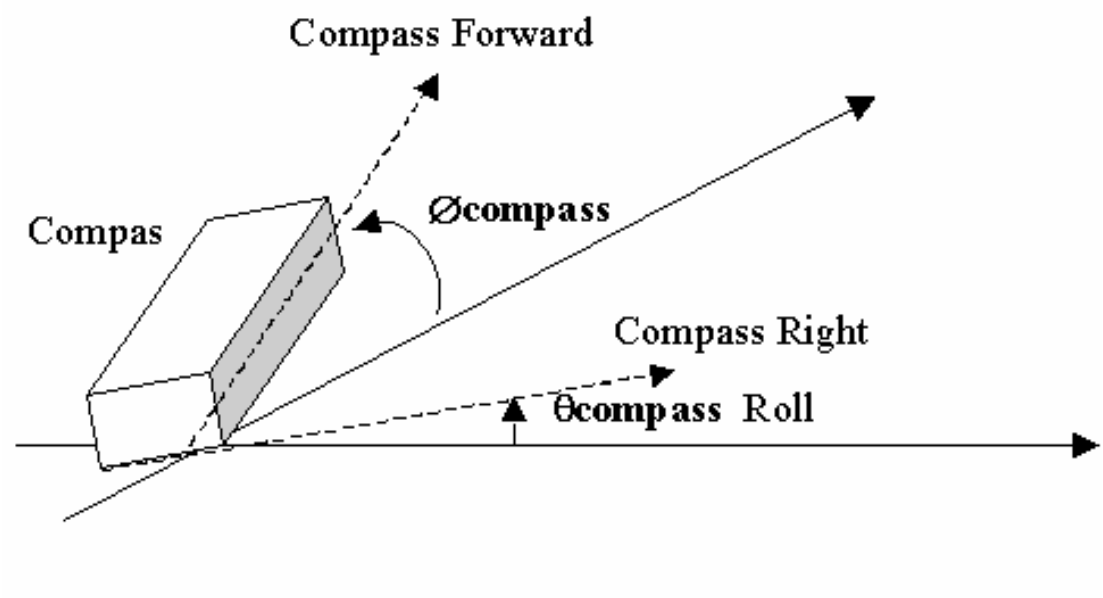


Figure 23: Compass orientation

The compensated horizontal magnetic values shall then be,

$$XH=Hx^* \cos(\varnothing_{compass})+Hy^*\sin(\theta_{compass})^*\sin(\varnothing_{compass})-Hz^*\cos(\theta_{compass})^*\sin(\varnothing_{compass})$$

Equation 4

$$YH=Hy^*\cos(\theta_{compass})+Hz^*\sin(\varnothing_{compass})$$

Equation 5

During the prototype, the first was obtained from a single axis magnetic sensor mounted perpendicular to the dual axis sensor used and the second was obtained with a dual-axis tilt sensor. Tilt compensated Azimuth heading was then calculated using Table 7 by replacing Hx and Hy with XH and YH respectively.

2.9.2 Compensating the Earth's Magnetic Field Distortion Due to Nearby Ferrous Material and Internal Offsets

After the X and Y magnetic field component had been compensated for the tilted orientation, the next step was to compensate the magnetic field

distortion caused by surrounding ferrous materials, such as substances in the PCB, electronics components, and nearby steel structures such as a barge or vessels. The two upper plots in Figure 24 show the actual distorted horizontal (tilt compensated) magnetic field reading from the prototype compass after it had been turn around for 360° near steel structures. As opposed to an ideal, non-distorted YH and XH plot that is a circle centered at the origin, it is clear that the magnetic readings were largely distorted. A software compensation technique was employed to rectify these situations based on the same application notes from Honeywell Inc.

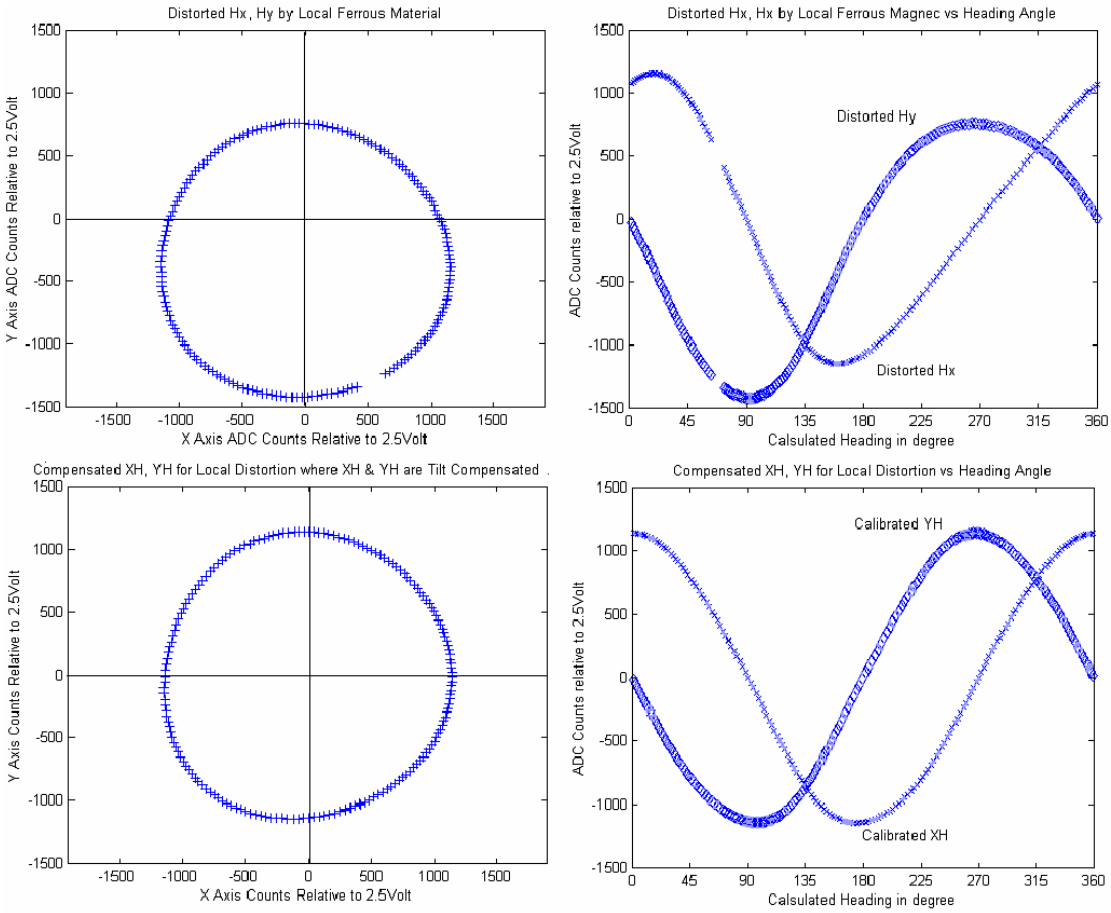


Figure 24: 360° Magnetic reading of prototype compass: ferrous interfered (top) and soft/hard iron compensated (bottom).

Two scaling factors (Xsf, Ysf) and two offset values (Xoff, Yoff) were introduced to respectively rectify the distortion of the circle and its offset from origin. The corrected value could be calculated as below,

XHc =Xsf* XH+Xoff,

Equation 6

Where XHc is corrected value (c denotes corrected) and XH are initially read (distorted) value.

YHc= Ysf* YH+ Yoff,

Equation 7

Where YHc is corrected value (c denotes corrected) and YH are initially read (distorted) value.

Once again the heading was then calculated using Table 7, but this time Hx and Hy were replaced with XHc and YHc respectively to obtain both tilt compensated and soft/hard iron compensated headings. These scaling factors and offsets could be obtained from the maximum and minimum values of the tilt compensated azimuth magnetic readings (XH and YH) by performing a 360° turn in the actual operating environment. Therefore, from the distorted values in Figure 24,

Xsf = 1 or (Ymax-Ymin)/Xmax/Xmin), which ever is greater

Equation 8

Ysf = 1 or (Xmax-Xmin)/(Ymax-Ymin), whichever is greater

Equation 9

Xoff = [(Xmax-Xmin)/2-Xmax]*Xsf

Equation 10

Yoff = [(Ymax-Ymin)/2-Ymax]*Ysf

Equation 11

The obtained Xsf, Ysf, Xoff and Yoff was then used to calculate XHc and YHc, which were then used to derive the azimuth headings, as shown in the two lower plots of Figure 24. The heading estimates are much better after the calibration; nevertheless, whenever there are changes in nearby ferrous disturbances, the compass had to be recalibrated.

Part of the residual differences could be due to offsets caused by thermal drift, offset of sensors networks, and DC offsets of analog electronics. In order to minimize these factors, a high current pulse of 1 ~ 2ms pulse width

was applied to the sensor to generate a large magnetic field that flipped the magnetization directions of the magnetic sensor. This approach worked because when the sensor polarity is flipped, the offset associated with the sensor bridges, on board electronics, as well as temperature drift will not be flipped. Therefore, adding the two reversed readings will cancel out the direction reading, leaving the offset value twice in magnitude, as below,

$$OS = (Vset + Vrst)/2$$

Equation 12

Where Vset is the immediate reading at one direction (SET) and Vrst is the reading at reversed direction (RESET)

The calculations were implemented with PC software after all the sensors have passed up their respective values. The prototype tests showed good repeatability with a tolerance of less than 0.5° at 90% of the time. Although the prototype showed relatively good repeatability, we were not able to calibrate the absolute heading to a precision that was satisfying. This was mainly due to the lack of precision fixture fabrication facility and high accuracy reference heading sensor during the heading calibration effort. The fixture manufactured was guaranteed to a tolerance of less then 5° between the mountings, while most COTS electronic compasses provided heading accuracy of 1° to 5°, which were not sufficient to facilitate calibration on our unit that in order to guarantee heading accuracy of less than 1°.

Towards the end of the project, an OEM module with the same magnetic and pitch and roll sensors was available on the market. This system is preferred to the prototype mainly because the entire calculations are done with its internal processor without needing processor resources from the PC104+; and secondly, the absolute heading tolerance is guaranteed to 0.5° @ ±40° pitch and roll angle. Some other reasons are that the OEM system is smaller in size and utilizes a smaller amount of power than the in-house prototype. This unit was then installed in a watertight housing and mounted on the hydrophone array with a RS232 connection to the HiDAQ.

2.10 Hydrophone Array

The successful determination of 3D directivity of ambient noise relied on the four omni-directional hydrophones that were positioned at the vertices of a tetrahedron to serve as a 3D sparse array. With three hydrophones positioned at the vertices of an equilateral triangle, we were able to resolve the direction of incoming signals by assuming that all snapping shrimp sit near the seabed and the sea surface reflected snapping shrimp clicks are 180° phase reverse to the direct signal. The system performance was improved by introducing the third hydrophone to form a tetrahedron in order to numerically identify the snap direction of the third axis and to increase the estimation accuracy.

2.10.1 Determining the Array Size

The four hydrophones could be arranged in various array sizes providing their cables are long enough. Nevertheless, the array size should be determined by finding a balance between the angular resolutions and the ability to deterministically identify a particular snapping shrimp snap across all four channels. For example, a larger array will provide better angular resolution, but an array that is too large would mean that the time needed for a signal to travel between two hydrophones could be too large such that multiple snaps existed in that time window, hence not able to classify snaps across channels easily. This section discusses the considerations made to determine the array size.

Since ambient noise is broadband, hydrophone separations are not restricted to less than half wavelength of the highest frequency of interest as compared to CW signals, which will produce grating lobes when the sensor separation exceeds the half wavelength criteria. Nevertheless, the hydrophone separation should be small enough so that the propagation delay between hydrophones at farthest point (which, in the worst case, is the length of the arm of tetrahedral, d) is kept less than inter-snap interval. Therefore the distance between hydrophones was determined by the frequency of

occurrence of the detectable transient signals (or snaps) existing in underwater ambient noise, which was estimated by the following calculation.

It is known that the estimated density of snapping shrimp snap could be around 0.1 to 0.01 snaps/second/meter^2 [30]. Therefore by estimating the total area where the snaps are detectable, the frequency of occurrence could be estimated and hence the maximum sensor distance. As shown in Figure 25, the farthest source distance, R, is taken as the distance where the spreading loss attenuates the snap source to a level that is undetectable by the analog electronics, i.e. when the signal after spreading loss is within the level of the analog's peak-to-peak noise floor. With an acquisition system of 12bit resolution, 10V peak-to-peak input voltage range, and 40mV worth of peak-to-peak noise (with 64dB analog amplification), we were left with 47dB dynamic range (we were able to acquire signals about 220 times larger than system noise without saturating). Assuming the system gain was set such that the nearest possible biological source clicks from seabed (4m directly below the tripod) were amplified to half (-6dB below) the dynamic range; the system would be able to identify any signals 6dB above the noise floor (without any signal processing gain); and, assuming the spreading loss was spherical (i.e. the spreading loss is 20log(R)), R (hence L) can be estimated to be about 100 meters, translating to an area of coverage to about 30,000 square meters.

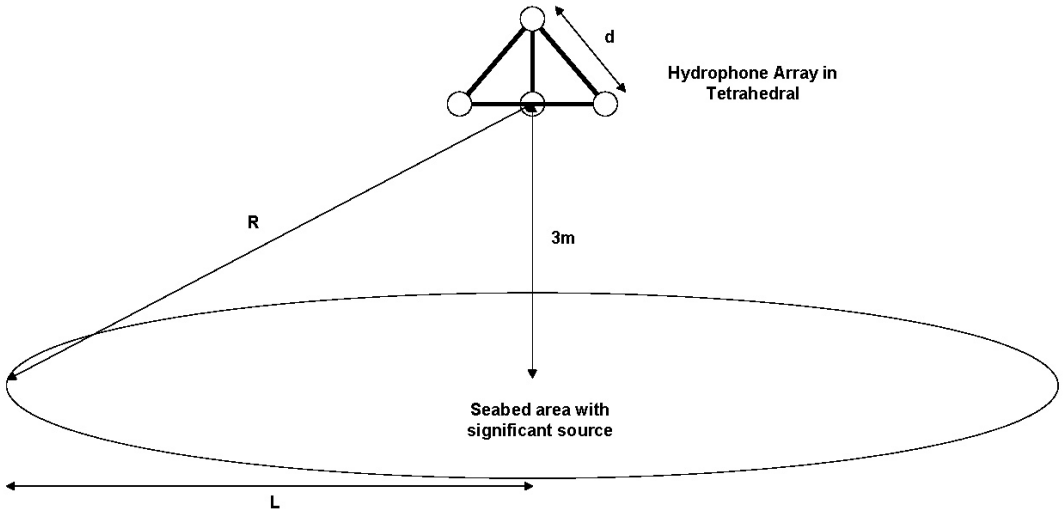


Figure 25: Area of interest and the distance between hydrophones.

Assuming the snap density of the area is about 0.01 snaps/second/meter^2, there will be roughly 300 snaps per second or 3.3 millisecond of average snap interval. Assuming nominal sound speed of 1540m/s in local seawater (as measured with a CTD), the average separation between snaps is about 5 meter. At areas where snapping shrimp noise density reaching 0.1 snaps/second/meter^2, the average separation between snaps could be as small as 0.5 meter.

Although the array aperture size could be as large as 5m according to estimation, the largest aperture size of array frame was kept around 1.1m (the distances between acoustic centers of the hydrophones are about 1.2m when mounted to the frame). This is because we do not need angular resolutions that are better than the accuracy of compass heading (which is around 0.5°). A smaller array size was helpful for portability and also provide safety factor of more than 4x (the array will work at places with snaps density of up to 0.04 snaps/second/meter^2).

2.10.2 Acoustically Transparent Mounting Frame

The four hydrophones were positioned at the corners of the tetrahedral frame mounted on a vertical rod. The material used in making this frame was chosen so that it did not distort the incoming waves. Two options were identified: one was to use plastic with an acoustic impedance close to sea water, and the other was to use a metal with diameter smaller than the wavelength of the highest frequency of interest so that minimum scattering was introduced to the incoming waves.

For the first option, the diameter of the plastic used needed to be relatively large (about 20mm) in order to provide enough strength. Therefore, it is important to use material with acoustic impedance that matches that of seawater so that it is transparent to sound waves. Since the off-the-shelf plastic materials locally available did not have the necessary acoustic impedance, it had to be calculated from alternative parameters through Equation 13 and Equation 14.

$Z = \rho \times Vp$

Equation 13

Where, Z is the acoustic impedance, kg/m^{2}s
ρ is the density of material, kg/m^3
Vp is the ”P” wave velocity, m/s

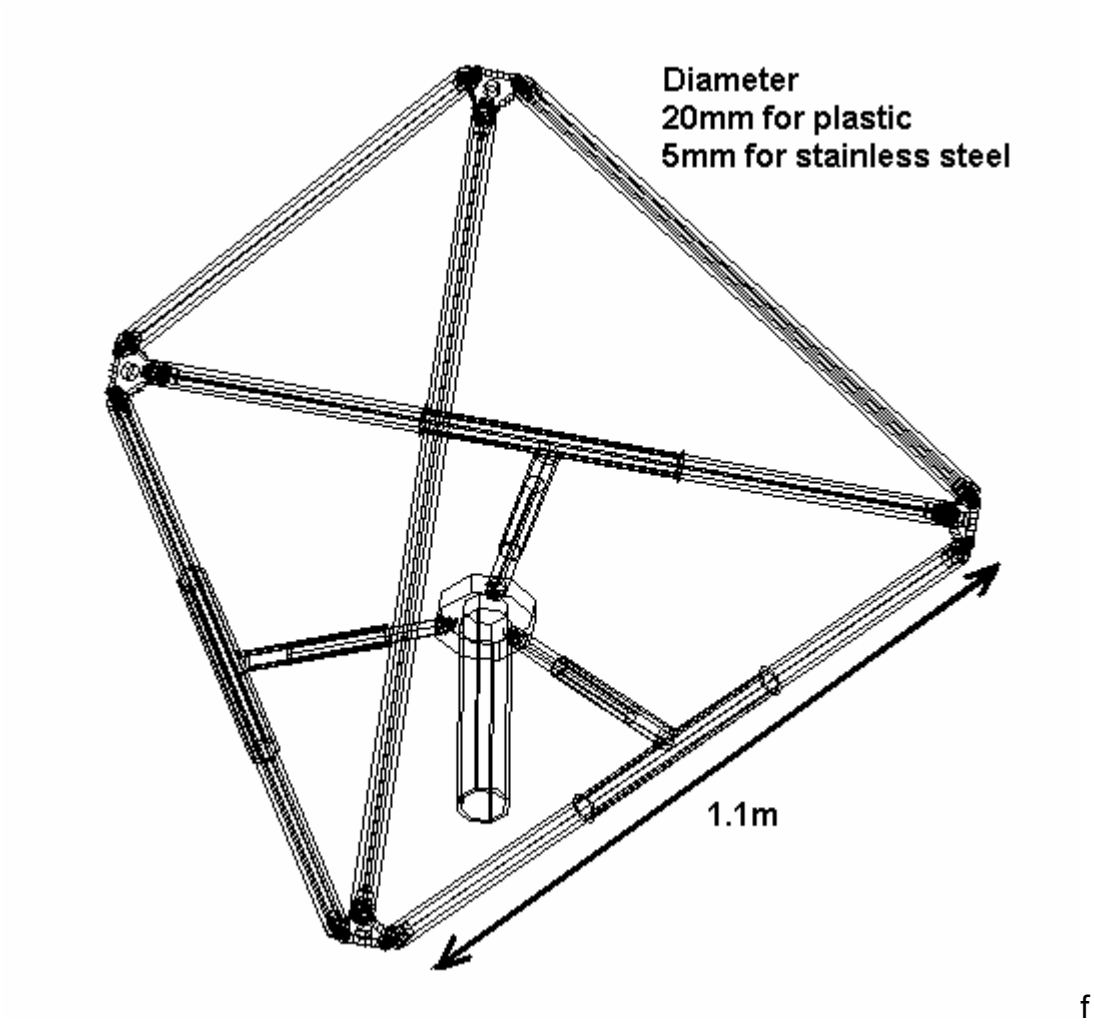
Where “P” wave velocity can be obtained from its relationship with Young Modulus as in Equation 14,

$$E = \frac{\rho V_p^{\,2}(1+\sigma)(1-2\sigma)}{(1-\sigma)} \quad , \qquad \textbf{Equation 14}$$

Where, ρ is the density of material, kg/m^3
σ is the Poisson coefficient,
Vp is the ”P” wave velocity, m/s

Young’s Modulus for different materials can be obtained relatively easily and provided us a good way to estimate the acoustic impedance of commercially available materials.

For the second option of using a metal rod, based on the highest frequency of interest (200kHz), and a sound speed of 1540m/s, the equivalent smallest wavelength is 7.7mm. Therefore, stainless steel 316 rods with 5mm diameter were selected to built the structure. Figure 26 shows an AutoCAD drawing of the frame.



f

Figure 26: Tetrahedral frame for the three-dimensional hydrophone array

2.11 Electronics Housing and Supporting Structure

This section discusses the mechanical design of the electronics housing and the supporting tripod. These housings and structures were mainly designed using Mechanical Desktop version 3 from AutoDesk.

3.11.1 Electronics Housing

An off the shelf PVC watertight housing manufactured by Prevco Inc. that was modified to our requirement, was initially used for packing the electronics. The housing was a low cost plastic design rated for use up to 100m water depths. Because of the space requirement, the housing was custom made to 46cm internal packaging length (the internal diameter remained unchanged at 17cm). The design entailed a threaded collar

securing mechanism; therefore no screwing or bolting was needed at the end caps to hold them in position; see Figure 27. The end caps were provided with piston O-ring seals to make the internal volume waterproof. The overall (external) length of the housing was 60cm with the maximum external diameter of 23cm, and weighed about 17kg in air. Provisions were made on the end caps to fit watertight connectors through which the sensors and the electronics could be accessed.

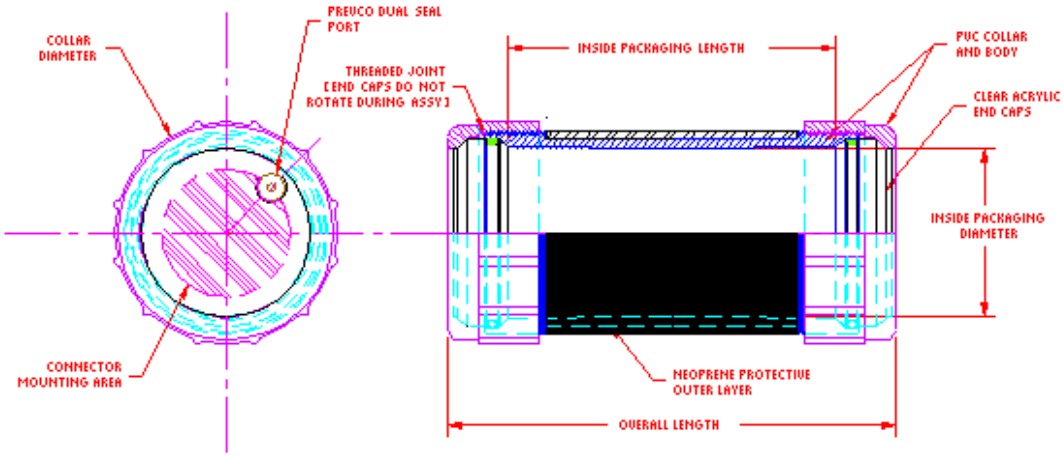


Figure 27: Underwater electronics housing (adapted from specification drawing, Prevco Inc.)

A special cylindrical cage was designed to mount the different electronic modules into a single electronics package that slotted into the housing. Because of the tight spacing constrain, the cage was built light as a holding structure rather than a strong mounting structure in order to keep the cage thickness and diameter of the supporting pillars small (see Figure 28). Since the internal cage was relatively weak, it was built such that it fit tightly into the underwater housing's internal space making use of the internal wall as main mechanical reinforcement. Important analog electronics were provided with shields to avoid any electromagnetic disturbance (EMI) caused by the internal processor clock and motor noise of the hard disks.

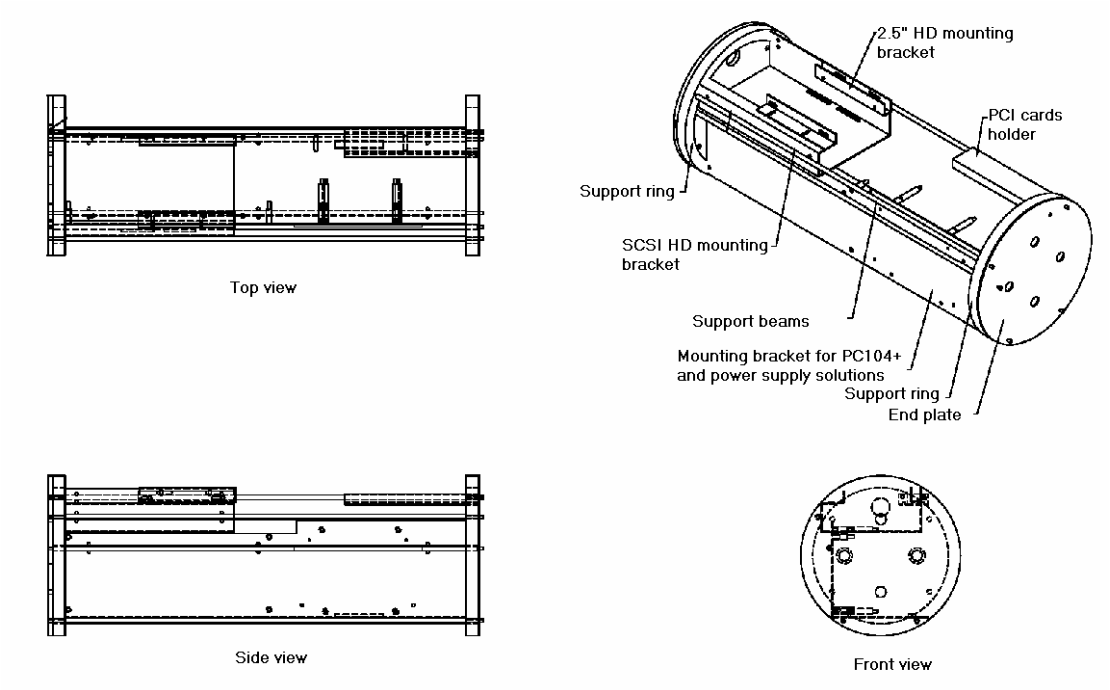


Figure 28: Mechanical drawing of the internal electronics cage and assembled electronics package

Heat dissipation issue was found to be a problem during field deployments; therefore, an aluminum cylinder was fabricated to replace the main PVC housing in order to provide better heat dissipation. This heat dissipation issue was mainly caused by some of the electronics module of the system (especially the high speed acquisition module and the 10,000-rpm SCSI hard disk), which generated enough heat that the temperature could reach 60ºC during the operation. The original end caps were kept because they would not contribute to the heat dissipation problem, as it was not in the main thermal path.

The new casing was also manufactured with some indents around the cylindrical body to provide a place for a clamp at each of the two ends to be mounted. Each of the mechanical clamps was fabricated to firmly grip the housing body at one of its end while provided a coupling at the other end. A series of fittings to the coupling were manufactured to provide various mounting possibilities that allowed the HiDAQ to be deployed in various configurations through these adaptors. Figure 29 shows the mechanical drawings and a picture of this new housing.

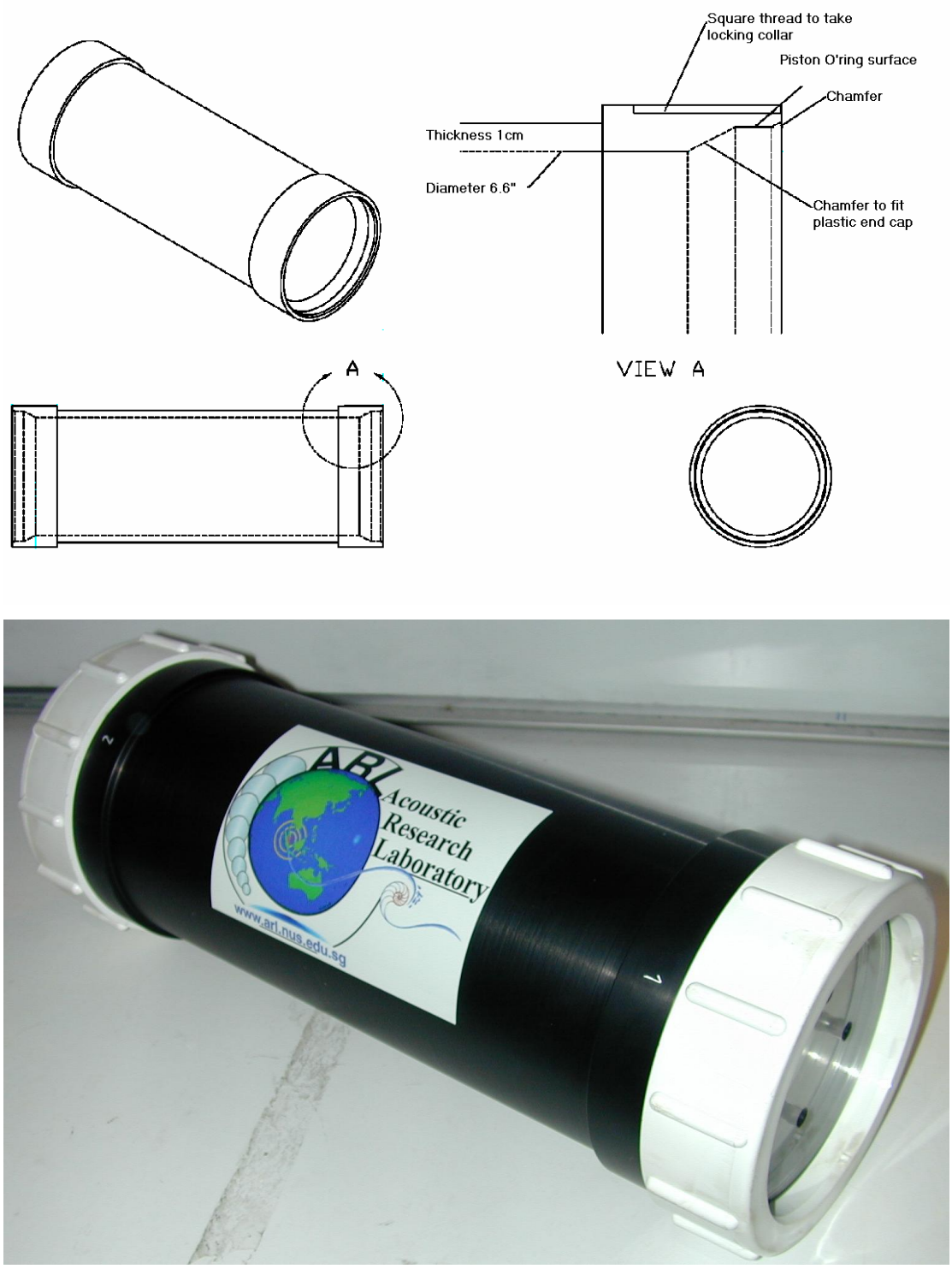


Figure 29: Mechanical drawing of the new underwater housing

3.11.2 Supporting Structure

In order to support the array at about 4m above the seabed, a telescopically adjustable stainless steel tripod was used. The structure was made modular to ease the process of site installation and transportation. The

tripod consisted of a main body, a vertical extension rod, leg extensions, and feet. The degree of leg openings was made adjustable to accommodate different drag forces at different sea conditions and seabed contour (30 degree leg opening for calm waters or 45 degree leg opening when current was stronger). The electronics housing and sensor array was designed such that it induced minimum drag at the top of the tripod and a 30° opening would most likely be sufficient in almost all cases.

The height of the tripod was adjustable from 2.3m to 4.5m and the lengths of the legs were also made adjustable in order to accommodate variations of the uneven seabed. This was accomplished by adjusting the telescopic coupling between the extensions legs and the main body, as well as the telescopic coupling between the vertical extension rod and the main body.

Feet were designed to have a large contact area with the seabed, to prevent the structure from sinking into a soft seabed composite such as silt or mud. The hydrophone array was then installed on top of the vertical extension rod while the electronics housing was attached below it.

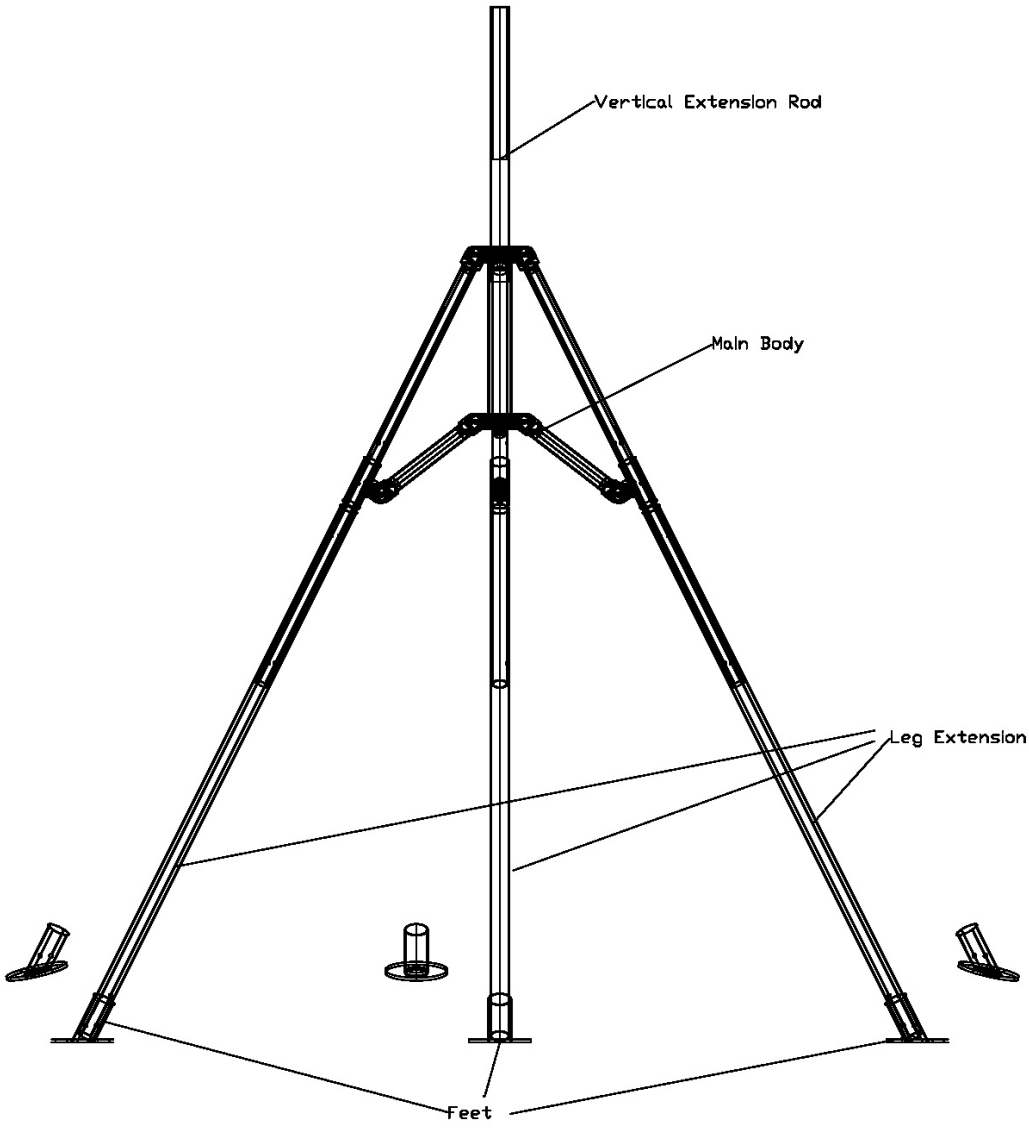


Figure 30: Mechanical drawing of the 4m tall stainless steel tripod.

CHAPTER 3
LABVIEW ACQUISITION SOFTWARE

The control and acquisition software was programmed using Labview®, a graphical programming language from National Instruments. Unlike conventional programming approaches, the software was 'drawn' using various block diagrams, symbols and connecting wires provided in the programming library. Each of the blocks represents a function with associated properties and operations. The drivers and controls to the acquisition hardware are provided (by the manufacturer) as an instrument block with various control interfaces. Figure 31 shows the program of the acquisition software used in the project. This standard program included modules for hardware interfaces, program controls, simple calculations and a graphical user interfaces (GUI).

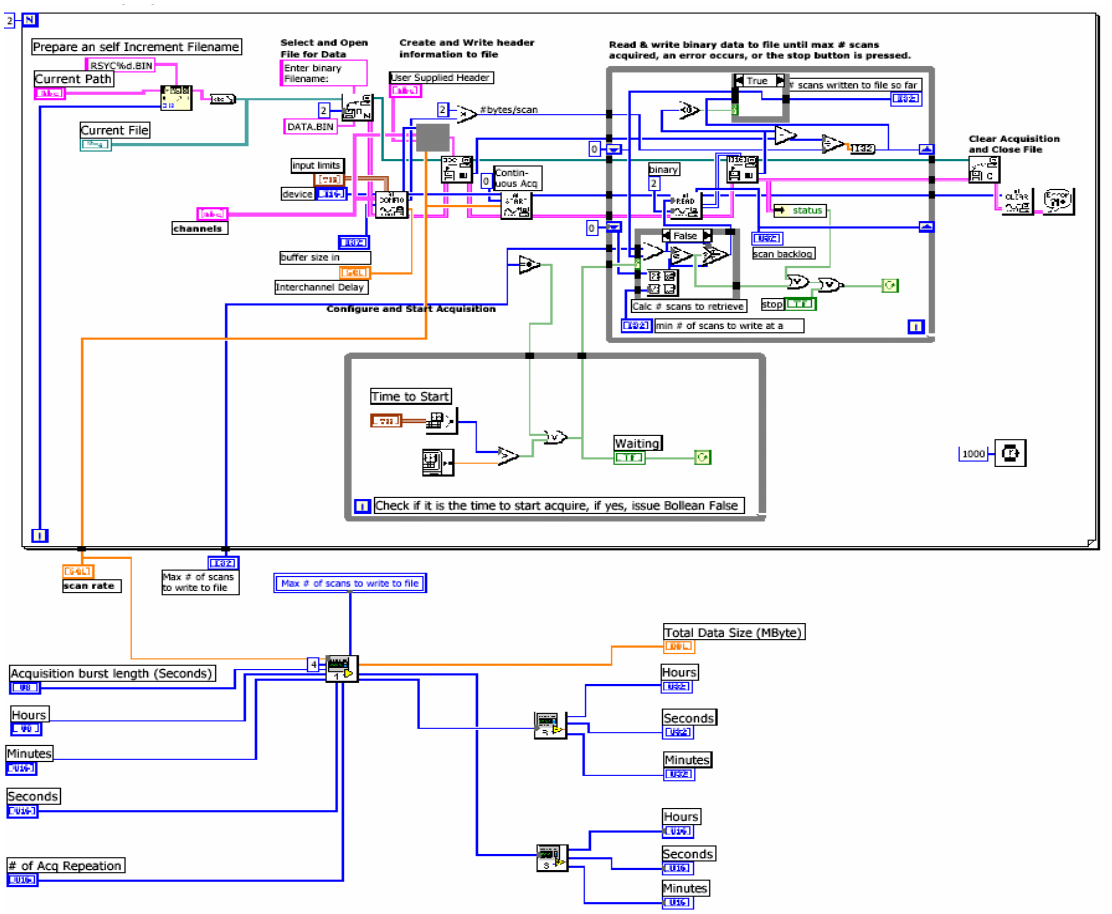


Figure 31: Diagram view of the acquisition software.

The control and acquisition software allowed the users to schedule the acquisition processes so that the data collection could be automated. The file names, and storage path could be specified in the program. It also allowed data acquisition to be carried out in sections with predetermined intervals. The following paragraphs explain the operation of the GUI.

Figure 32 shows the GUI front end of the data acquisition software. Labels 1 to 5 mark the controls of the acquisition hardware. Item 1 allows the user to specify the number of channels to be activated for the operation; in this case, all four channels are activated. Both items 2 and 9 specify the length of each acquisition event, the first is in terms of number of samples while the later is in seconds. Item 3 is a panel to control the sampling rate, the size of the memory buffer to allocate and the size of a data block to transfer into the hard disk each time. The combination of these parameters will determine the performance of the acquisition. Item 4 provides the user with a mean to control the voltage range of each channel and the inter-channel sampling delay. This is useful for optimizing the usage of the dynamic range of the analog to digital converter. Item 5 is a toggle button to activate and stop the entire acquisition processes. Item 6 is a display box that provides feedbacks of the current acquisition process: the number of samples successfully recorded and the occupied buffer space at any time. These two numbers give a good indication of how optimized the acquisition parameters are at anytime.

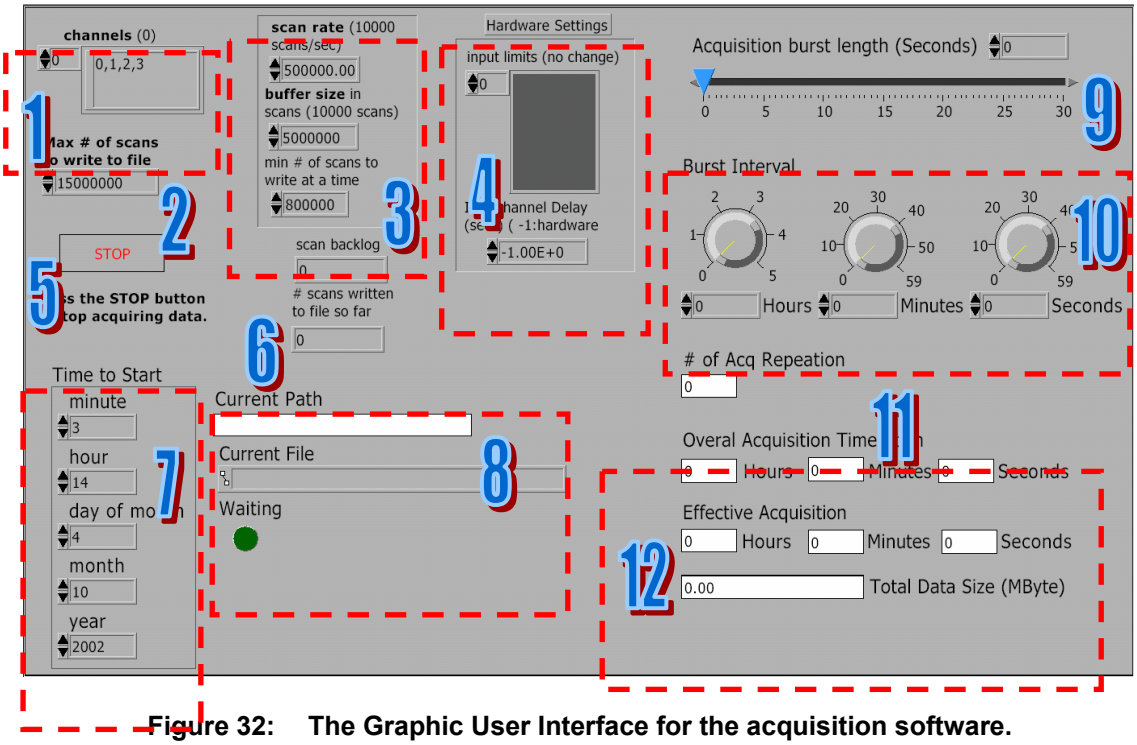


Figure 32: The Graphic User Interface for the acquisition software.

The software also allows the user to schedule a start time for the whole acquisition process; this could be configured with the input boxes at item 7. Item 8 allows the user to select the filename and folder location where the acquired data would be saved. If the data collection is programmed as multiple successive acquisitions, the filename can be suffixed by an index number, according to the sequences. The LED labeled 'Waiting' will light during the idle periods between these multiple acquisitions, as well as when waiting for the scheduled starting time of the acquisition process as programmed in item 7.

The next section of the software allowed the user to program multiple acquisitions, and the length of these acquisitions. Item-9 is a slide bar to adjust the duration of each acquisition burst. Item 10 determines the length of intervals between acquisitions, while the number of repetition is programmable through item 11. After all these have been set, the displays in item 12 will give a summary of the overall time span of the entire process, the total harddisk size needed and its equivalent amount of data in terms of time of continuous recording.

CHAPTER 4
BEAMFORMING ALGORITHM

This section describes the beamforming algorithm used by the array to determine the direction of the arrival of transient signals. The data analysis could be done in two ways: firstly by evaluating the energy of each direction in 3 dimensions by adjusting the delay of time series of each direction and to add them together; the second was by deterministically finding the individual clicks on all four channels and to estimate their directions from the delay. The first would take up a lot of processing power as the entire time series had to be repeatedly calculated in each three dimensional directions at the desired resolution. The second method on the other hand will only process sections of the time series that have transients and thus saves processing time. Since we were looking at snapping shrimp clicks, which are broadband and transient in nature, the second method proved to be a more efficient choice. The following sections describe the algorithm of the second method and its geometry.

4.1 Time Difference of Arrival (TDOA) Beamforming

The recorded time series of one of the channels was first scanned through to look for transients, each of them was then used as a template to search through the other three channels within a defined time window based on the size of the array. Once all the clicks were identified, their inter-channel time delays were calculated. Based on these delays, the direction of arrival of each of the clicks was then estimated.

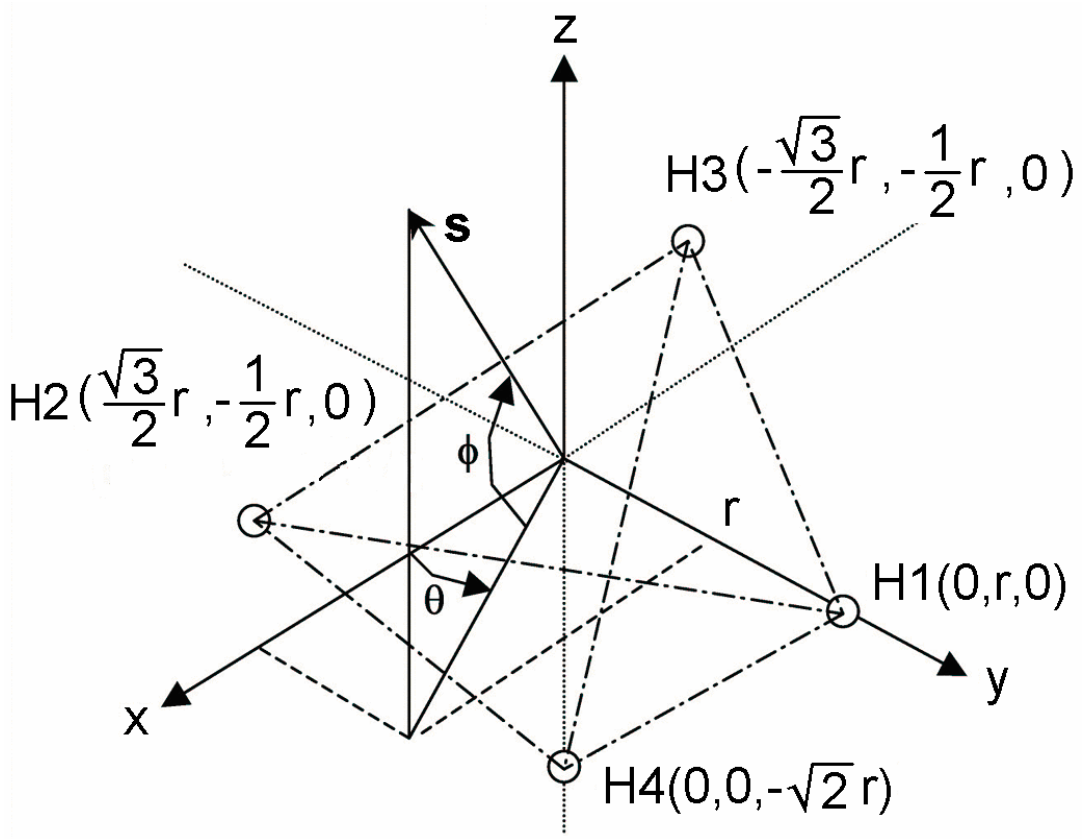


Figure 33: Geometry of the tetrahedral array

Referring to a coordinate system originating at the centroid of the triangle that forms the base of tetrahedral and having the forth hydrophone (H4) pointed down (see Figure 33), the array geometry is described in vectors as $\underline{h}_1$, $\underline{h}_2$, $\underline{h}_3$, and $\underline{h}_4$ representing to the positions of hydrophones H1, H2, H3 and H4. The distance between the tip hydrophone (H1, or H2, or H3) of the base triangular and the origin, r, can be related to tetrahedron arm length, *l*, as

$$r = \frac{l}{\sqrt{3}},$$

Equation 15

Where *l* is the distance between hydrophones, which was 1.2 m in the current setup.

The direction of an incoming wave can be described by a unit vector $\underline{s}$ in Cartesian axis as shown in Figure 33 and expressed in form of matrix, given by Equation 16,

$$\underline{s} = \begin{bmatrix} -\cos\theta\cos\phi \\ -\cos\phi\sin\theta \\ -\sin\theta \end{bmatrix}, \qquad \textbf{Equation 16}$$

By taking the dot product of the geometry vector of the hydrophone locations and the vector of the incoming sound wave, the effective distance of each hydrophone from the origin, projected into the direction of incoming wave was obtained,

$$d_i = \underline{h}_i \bullet \underline{s}, \qquad \textbf{Equation 17}$$

where i = 1, 2, 3, 4 for the four hydrophones.

Therefore, the travel time delays in terms of distance between hydrophone H2, H3 and H4 with reference to hydrophone H1 can be re-written as

$$\begin{aligned} D_{j1} &= (d_j - d_1) \\ &= (\underline{h}_j - \underline{h}_1) \bullet \underline{s} \end{aligned} \qquad \textbf{Equation 18}$$

where j = 2, 3, 4 for the hydrophones.

By taking into account the speed of sound in water and the sampling rate of HiDAQ, the time lags between channels in term of sampling interval are therefore can be written as

$$T_{j1} = \frac{fs}{c}\left[(\underline{h}_j - \underline{h}_1) \bullet \underline{s}\right] \qquad \textbf{Equation 19}$$

T_{j1} can be obtained from the recorded time series when the inter-channel time delay is calculated after each snap has been identified in all four channels. $\underline{h}_j$ and $\underline{h}_1$ can be obtained from the geometry of the array; fs, the sampling frequency of each individual channel, which was set to 500kSa/s; and c, the sea water sound velocity at the site, was measured to be 1540 m/s using a CTD. Therefore, the unit vector of incoming acoustic wave, s(x,y,z), was solved in term of x, y and z axis (from which we obtain the θ and ϕ later).

CHAPTER 5
EXPERIMENT SETUP

The system, consisting of the HiDAQ module and the tetrahedral array, could be deployed in various configurations depending on how the modules were mounted and coupled. It could be deployed form the surface or as a bottom mounted system, each either in standalone mode or with a cable attached.

For the purpose of snapping shrimp distribution estimation, we deployed the system in three different configurations. The first two configurations deployed the system from surface platforms, attached to either a buoy or to a barge or vessel. For the first option, the tetrahedral frame was coupled directly to the electronics housing and the complete module was attached to a custom-made flexible spar-buoy that minimized the vertical oscillation caused by surface waves. This configuration ran in stand-alone mode and allowed us to deploy the system in the open sea without a surface vessel near by. The same physical setup was also deployed from a barge (or at times from surface vessels); it was secured from the surface by tensioned ropes, thus avoiding excessive rotational oscillations. The tetrahedral frame was deployed as in the geometry orientation in Figure 33 for these two cases, typically 10 to 17 meter from the seabed, as illustrated in Figure 34. With these configurations, we were able to map more than 30,000m^2 of area centered at the array.

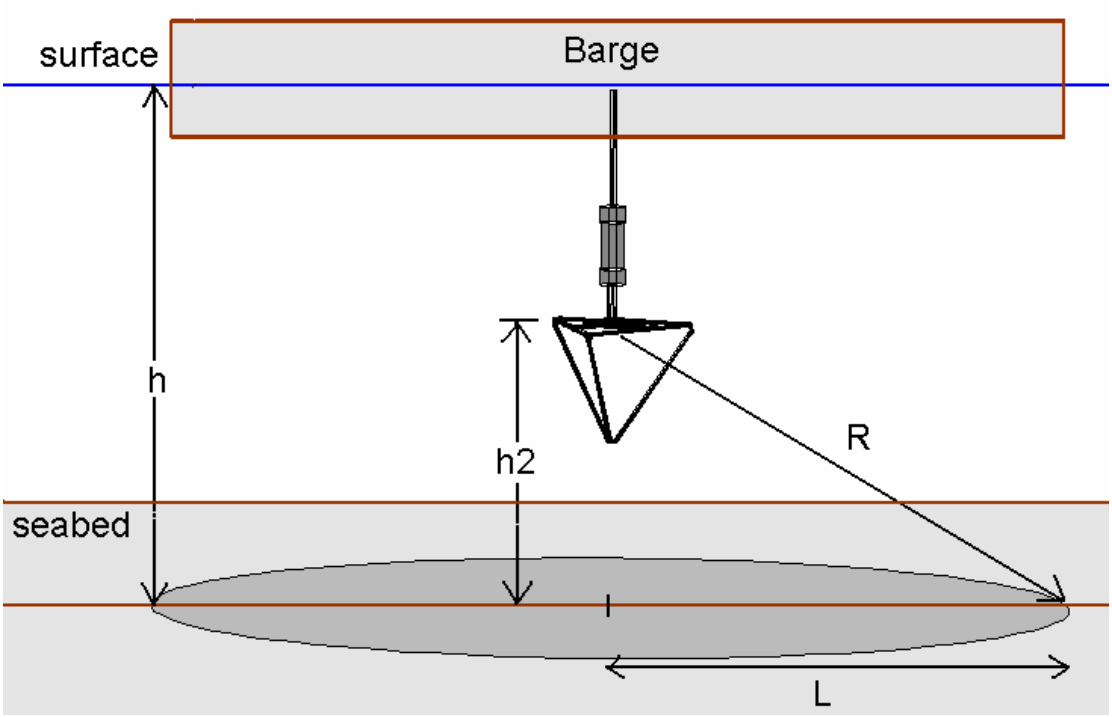


Figure 34: HiDAQ in surface mounted configuration.

A number of deployments were also been done in a bottom-mounted configuration, with the array attached on top of the 4-meter tall tripod. The entire system was placed on the seabed as shown in Figure 35. The electronics housing was attached at the lower end of vertical pole using a customized fitting. With the tetrahedral array mounted 4 meters above the seabed, we were able to map more than 20,000m^2 of area centered at the tripod.

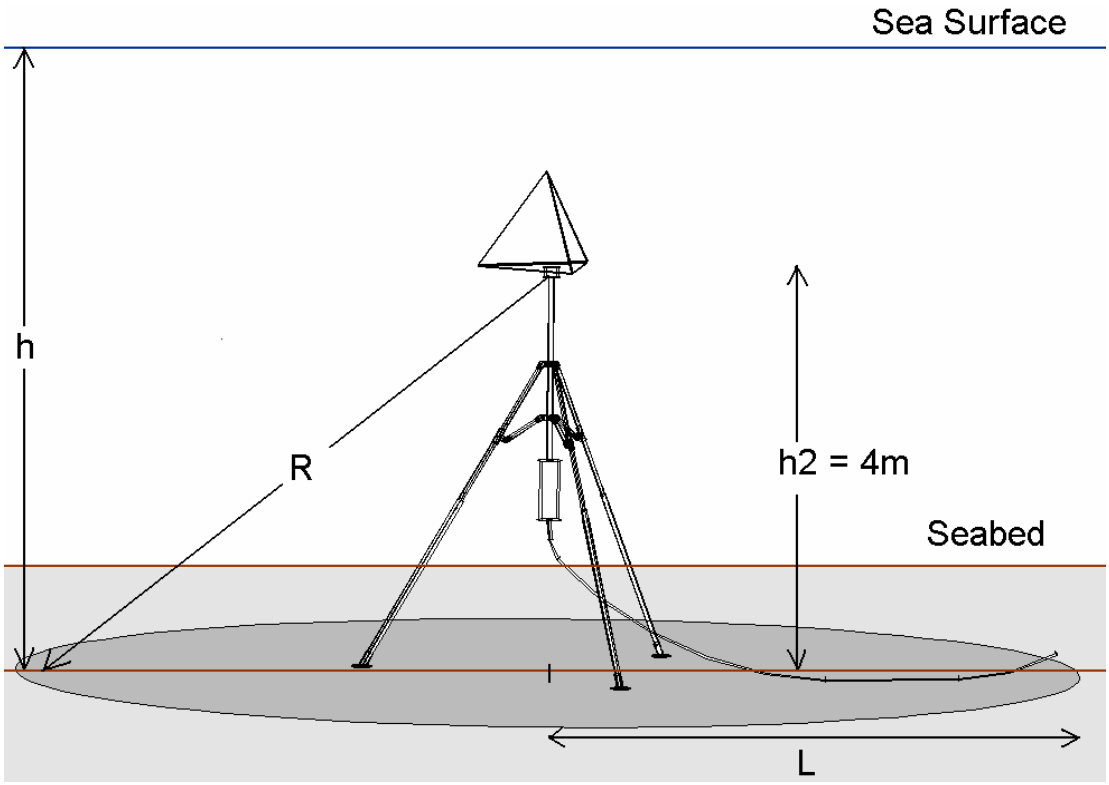


Figure 35: HiDAQ in bottom-mounted configuration.

Deployment of the surface mounted configuration was a relatively straightforward task since everything could be done from the surface. The picture on the right in Figure 1 shows the array and the electronic package hanging from a crane similar to the deployment form a surface flotation. On the other hand, deployment in the bottom-mounted configuration was more complicated due to the size of the tripod. Diver support was required to first install the tripod on the seabed, followed by the installation of the tetrahedral frame and the electronics package. The picture on the left in Figure 1 shows a setup with half of the tripod excluding the full legs and vertical extension rod; the electronic housing would then mounted at the bottom end of the vertical rod.

5.1 Mapping Noise Sources at the Seabed

With the estimated θ and ϕ, the measured water depth h, and the height of array from seabed h2, we could estimate the spatial distribution of the sources if we assume that the seabed is flat and that the sources are located on the seabed. We can also estimate the ranges of each identified

transient source from the array, R. With the known R, the spreading loss for the range can be corrected and the source level of the transients can be estimated.

The radius of the area mapped, L, is determined by the statistics of the source strengths of the local area ambient noise, the acquisition's system noise performance, the array size and the array height from seabed. The first two determine how far away from the array before a source cannot be detected due to system noise; while the last two determine how far away from the array before the range estimation tolerance become too large that source level estimation could be meaningless. The signal processing was implemented with the algorithms described in Chapter 4 using Matlab scripts.

5.2 Source Level Estimation Tolerance

The algorithm assumes a flat seabed and that the entire snapping shrimps population stays on the seabed. Nevertheless, that might not be true in the actual scenario in which the seabed might have some local variations, and the snapping shrimps might stay near to seabed instead of on it and the cavitation bubbles produced by the snapping shrimps could collapse at different heights off of the seabed (although their height variations are small). Referring to Figure 36, the source level estimations might include errors introduced by the error in range estimation (δR) if snapping shrimp snaps goes off at a height (δh) from the seabed (i.e. at location P2 rather than P1).

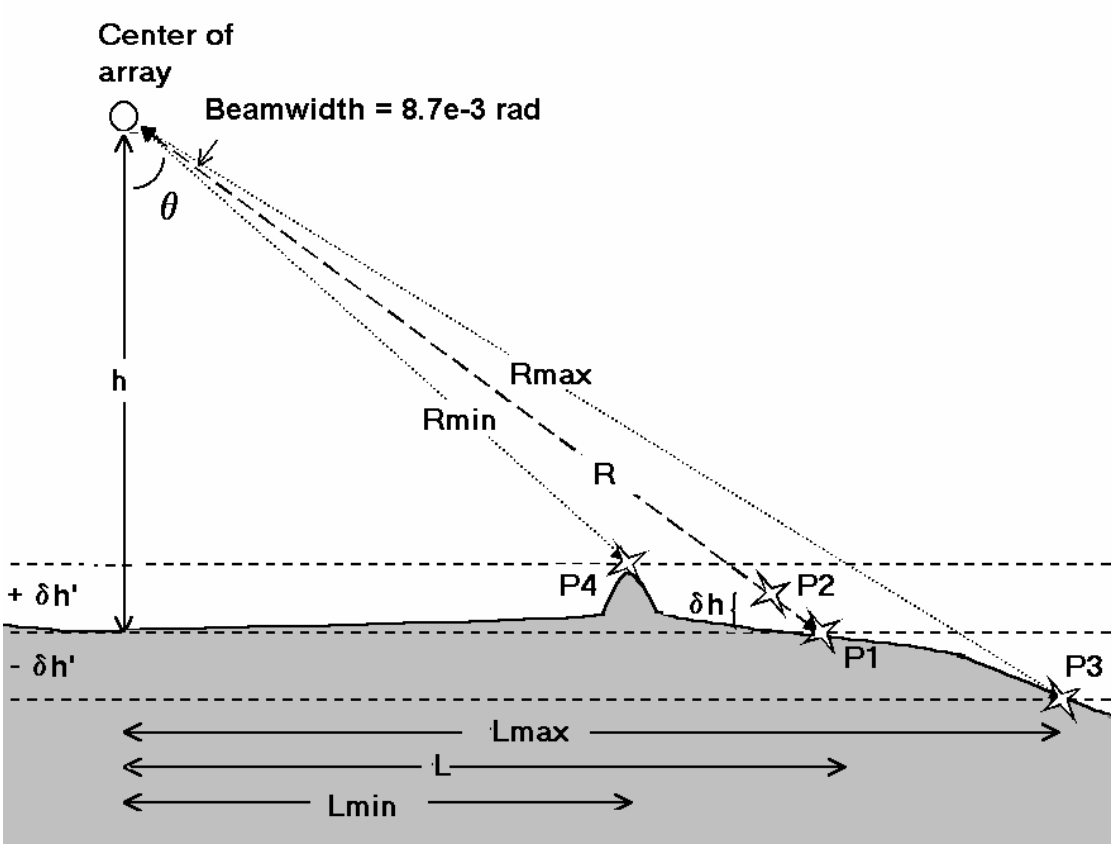


Figure 36: Range estimation errors due to snapping position and seabed variation

The relation between δR and δh is dependent on the height of the array from the seabed and the horizontal distance of the source from the array. The ratio between the range (of the center of the beam) estimation error and the source height tolerance is represented by Equation 20. The accuracy of range estimation would become worse when h becomes small or L becomes large. Therefore, the array height was kept as high as possible during the trials. At the worst-case scenario δR will be about $25\delta h$ where the system is bottom mounted (h = 4m) and the snapping snap is at the farthest and yet detectable range (L is about 100m).

$$\frac{\delta R}{\delta h} = \sqrt{1 + \left(\frac{L}{h}\right)^2}$$

Equation 20

Apart from this, as the array has a beamwidth of about 0.5° at 150kHz, by assuming the snaps detected are within the footprint of the beamwidth, the snap could gone off at any point between P3 and P4. As the range error is

more significant when L is larger, it is safe to assume that Rmax-R is larger than R-Rmin and take the earlier as the upper bound of the estimation error.

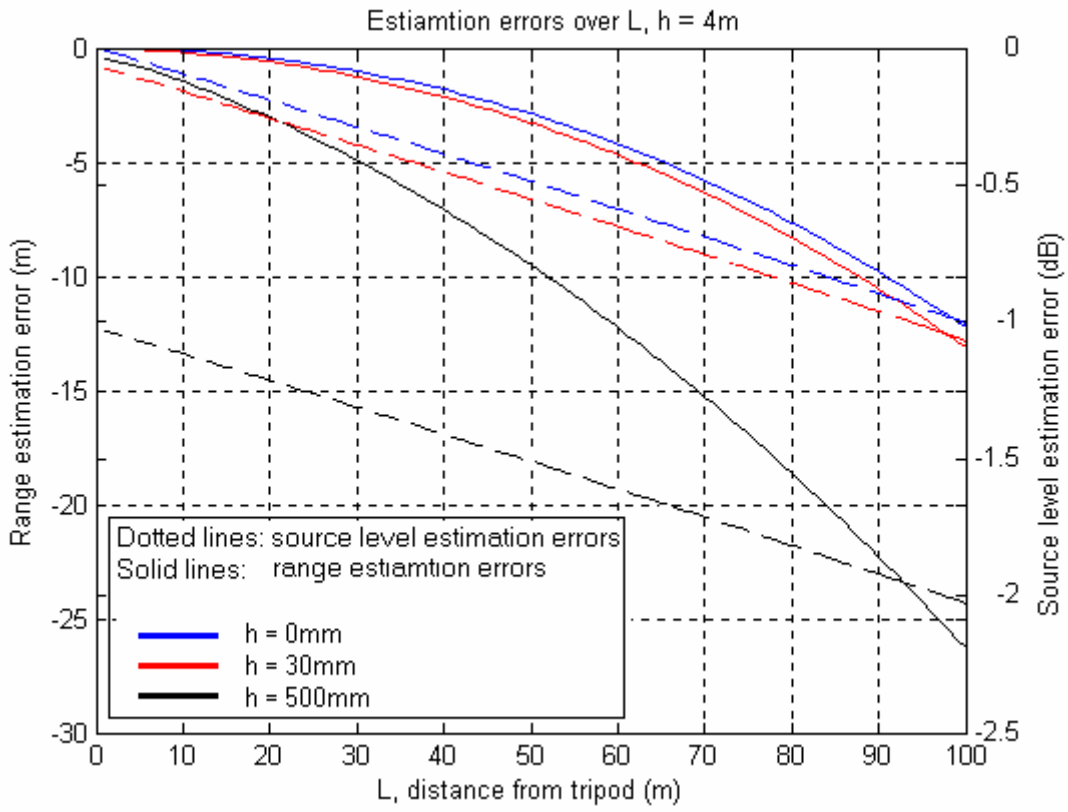


Figure 37: Range estimation error over source distance from tripod

By taking the above considerations, the worst-case source level estimation error would not be more than ±1.8dB when the vertical position of the source is ±500mm (mainly due to the local seabed variation) at 80-meter distance from the bottom-mounted configuration. The source level estimation errors caused by localize seabed variation could be corrected if the bottom bathymetry is known, leaving the estimation error of less than 0.9dB due to the snapping shrimp bubble collapse height (around 3cm) at source is 80 meters away.

5.3 Simulation

Based on the geometry described in chapter 4, a series of Matlab scripts have been coded to perform the signal processing. A simple simulation was done to verify the geometry mathematics and the overall code functionality. In the simulation, typical snapping shrimp snaps with 190dB re

1Pa @ 1m peak-to-peak source level were added into a time series at positions that corresponded to the inter-channel delays as if the shrimps had been snapping form a sets of locations and angles. Random noises of –26 dB below the snapping shrimp signal had been added in all four channels. Snapping shrimp snaps arranged in two rings on seabed with different diameters at 0.5° intervals were simulated. These simulated time series were then processed with the beamforming algorithm to test its accuracy. The results of the circle simulation are shown in Figure 38. The result shows that the source level estimation error is within ±0.2dB at ideal situation (seabed is flat and shrimp is on the seabed) when the sources are about 22 meters from the array, a number that is within the prediction of the discussion in section 5.2.

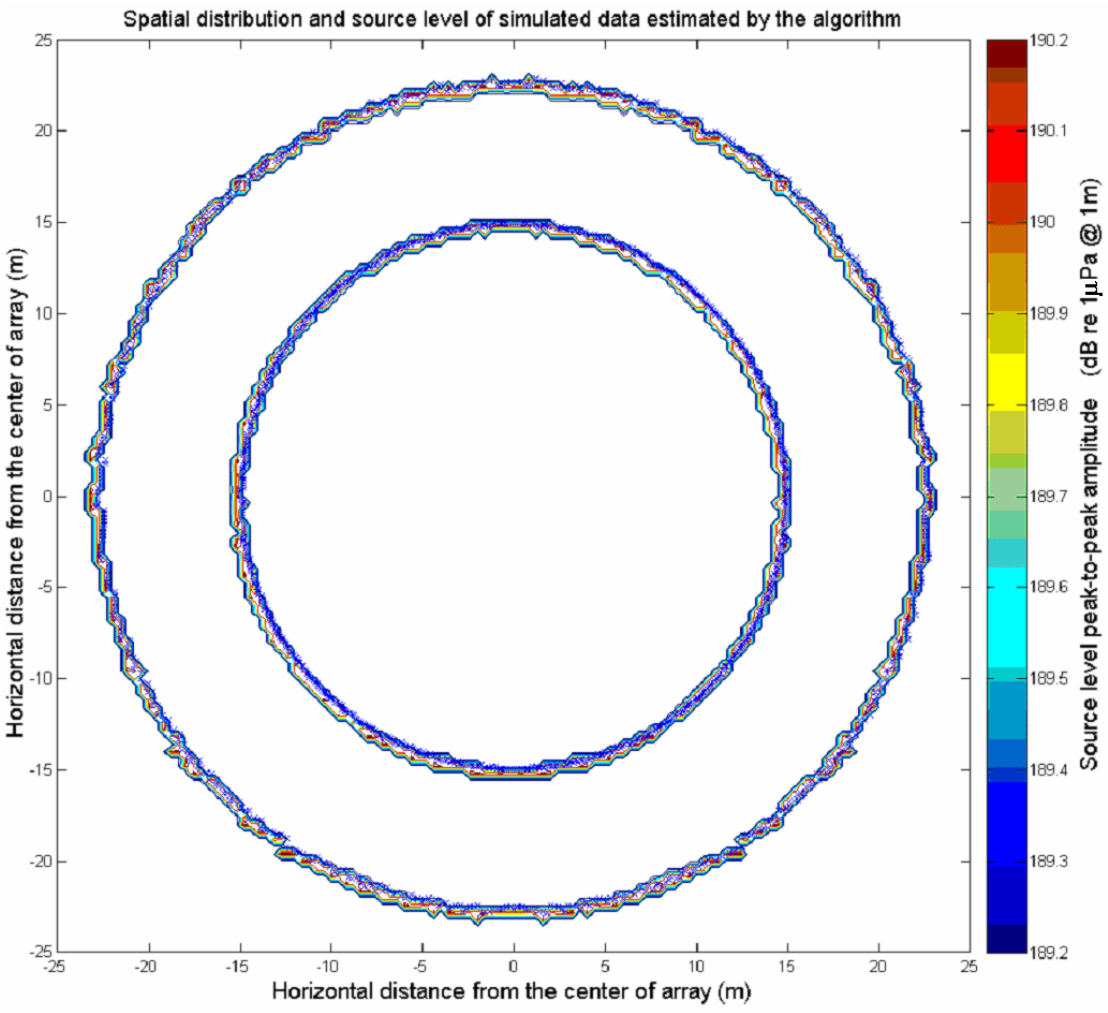


Figure 38: Inverted source map from a simulated distribution

CHAPTER 6
FIELD EXPERIMENTS

A number of field experiments were carried out beginning April 2002. The main test areas were the local waters around the Southern Islands. All three of the above mentioned deployment configurations were tested at the various sites. Modifications to the system (such as the mechanical mounting, housing robustness, software and electronics) have been carried out based on the experiences of these field trips, which helped in evolving the system towards robust and stable equipment as of now.

The first deployment was using a flexible spar-buoy. HiDAQ was configured in stand-alone operation mode and set to acquire data at a predetermined time. The deployment was done from a small 28-foot aluminum boat with three personnel on board. The system was assembled and configured at harbor, transported to the site and deployed. The actual deployment process was done in 10 minutes once the boat was at the deployment site (see Figure 39). The disadvantages of this type of a deployment were that the array was subjected to both translational and rotational oscillations caused by waves on the sea surface. The oscillations were tolerated after precaution measurements have been deployed, as the sound speed on which the snaps were traveling is much faster than the movement of the spar buoy. In addition, the rotational oscillation was corrected by keeping track of the array heading during the acquisition. The vertical oscillations of the first issue was minimized by using a spar-buoy, which have a small cross section diameter that makes it less susceptible to buoyancy changes caused by a surface wave, hence making it more stable vertically. Deployments using spar buoy were useful for quick, short-duration (less than 5 hours) data acquisitions that were limited by the battery capacity with the tolerance of a higher estimation error than bottom mounted configuration.

Figure 39: Deployment of HiDAQ using a tubular spar-buoy

A number of deployments were carried out using the bottom-mounted system at Raffles Anchorage using the 4-meter tall stainless steel tripod. Weighing about 85kg when fully assembled, the tripod was deployed with the help of a surface crane and diver teams. Figure 40 shows different stages of the deployment: (1) the tripod was firstly assembled at surface, and was then lowered to seabed (2) with the help of a crane, lifting bags and diver support. Once the tripod was installed on the seabed, the hydrophone array (3) and the electronic package (4) were then brought down by the divers on separate dives, which were then assembled on the tripod. After the structures have been deployed, the 50-meter underwater cable was then attached. Figure 41 shows the host computer at the surface, which can be simply any PC system with Ethernet connection and Microsoft Netmeeting software installed. Although the deployment of the tripod involved heavy jobs, it has been successfully deployed using a 38-foot aluminum boat with onboard crane support on several occasions.

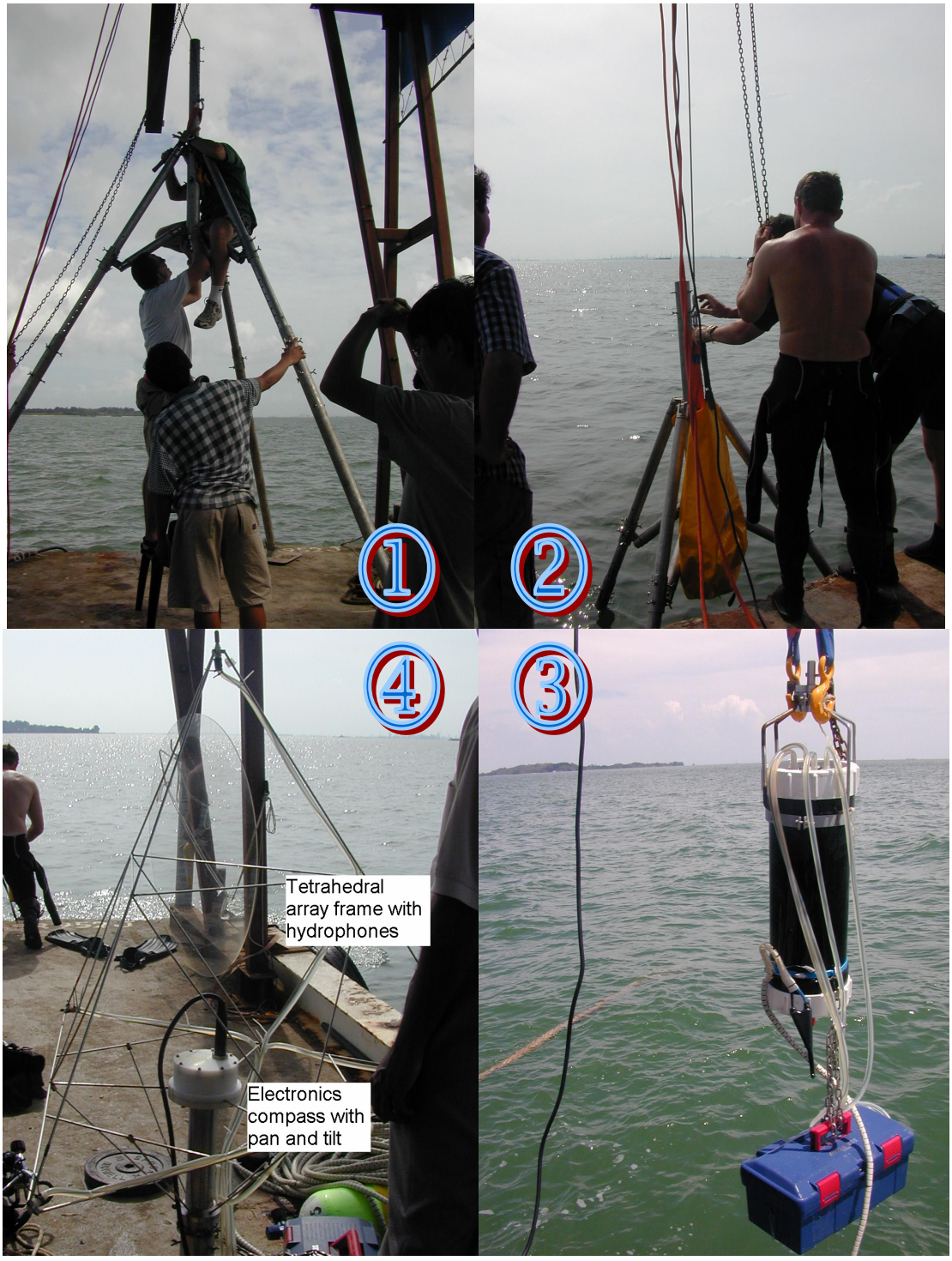


Figure 40: Deployment of HiDAQ in bottom-mounted configuration from a barge

Figure 41: The remote control station: a simple laptop with Ethernet connection.

CHAPTER 7
RESULTS

The results to be discussed in the following sections are based on data sets collected from 3 field trips at different sites: One is at Selat Pauh (off Pulau Hantu) and the other two are data sets collected from Raffles Anchorage on two different occasions and locations. All sites exhibit nominal depth of 15 to 20 meters and are near to reef patches. The Selat Pauh area has a mixed bottom type from silt/mud to sand toward south. The first two sets of data were collected with HiDAQ deployed from a barge anchored at both areas on separate occasions. The third data set was collected from a bottom-mounted deployment using the 4-meter tall tripod at Raffles Anchorage. The data from Selat Pauh was collected during daytime around 13:15~13:35 hours with the surface mounted HiDAQ. The data was acquired in multiples of 30 second continuous data separated at 10 second idle, with overall 15 minutes of data. The first data set from Raffles Reserve were taken early in the morning between 2:40 ~ 3:00, with a total of about 20 minutes worth of data collected in multiple 30 second files. On the other hand, the second set of data from Raffles Reserve was acquired from 16:30 in the afternoon thru the night until 06:18 in the next day's morning. The array was deployed in bottom-mounted configuration supported by a tripod. The automated acquisition was programmed such that it recorded 2.5 minutes of data every 2 to 3 hours. Due to the long acquisition period, it was AC-powered and remotely controlled from a barge about 10 meters away.

The sea floor was assumed flat in the analysis, even though the area could have some small depth variations. We also assumed that the sound speed in water was 1540m/s and stayed constant over the data acquisition period within the entire area. The results have been published at MTS/IEEE conference [3]. Square of measured acoustic pressure is used as an indication of the acoustic power through out the text whenever power is discussed although the actual acoustic power is measured as pressure squared divide by acoustic impedance (density multiply by the speed of sound

in the medium). This is possible because the water density and sound speed in the waters of our experiment site are almost constant due to their shallowness in depth (< 25m).

7.1 Power Distribution Function of Local Ambient Noise

A 30 second long time series was selected from the recordings of each site. The frequency band of the recorded time series were low pass filtered digitally at 180kHz with a 10^{th} order Elliptic filter during signal processing. This is to remove a high frequency sonar pings with center frequency around 200kHz. The median and standard deviation of the power distribution at Selat Pauh were estimated to be 3.48×10^{14} μPa^2 and 1.48×10^{14} μPa^2; while 5.21×10^{14} μPa^2 and 1.45×10^{14} μPa^2 respectively at Raffles Reserve.

Each time series was divided into 8msec time slices, from which the power in each window was calculated to form a vector of power at 8msec bins. The Probability Density Function (PDF) of the power distribution was then plotted. The distribution showed a significant skew that approximates lognormal distribution, as observe by previous studies in Singaporean waters, which in turn suggest a hypothesis that it could be caused by noise sources that are either temporally homogeneous but spatially clustered distribution or temporally clustered but spatially homogeneous distribution [15]. Nevertheless both distributions failed statistical test for lognormal distribution. In order to investigate further, theoretical lognormal PDF curves were calculated and plotted on top of the distributions obtained from field trip, as seen in Figure 42. The theoretical curves were generated by estimating the parameters of the best-fit normal distribution of the natural logarithm of the measured distributions.

First, the lognormal parameters μ (mean of the natural-log of the power distribution) and σ (standard deviation of the natural-log of the power distribution) of the power distribution were estimated using 'lognfit' function in Matlab. These estimated parameters were then used as initial values to manually find the best lognormal PDF fit to the acquired distribution. Lognormal distribution PDF functions were then generated based on the

manually iterated parameters and plotted on top of the distribution of the original power vector. The μ of a good approximate lognormal distribution fits are natural log of 3.3×10^{14} μPa^2 at Selat Pauh and natural log of 4.9×10^{14} μPa^2 at Raffles Reserve while the σ are approximated to 0.3 and 0.19 respectively. Note that these parameters shall not be mistaken as the mean and the standard deviation of the power distributions.

The collected distribution shows deviation from the lognormal fit and could be due to the presents of several unnatural sounds in the data set, such as a tonal around 58kHz, depth sounder pings around 38kHz and 200kHz. The sonar pings at 200kHz was removed with a low pass filter but the 58kHz tonal and 38kHz pings (around 0.9 seconds interval) were left in the data and could have changed the shape of the distributions. The lower end of the distribution seems to be missing compared to the theoretical curve, this could be caused by the system noise that limits the power to always above a number at any time.

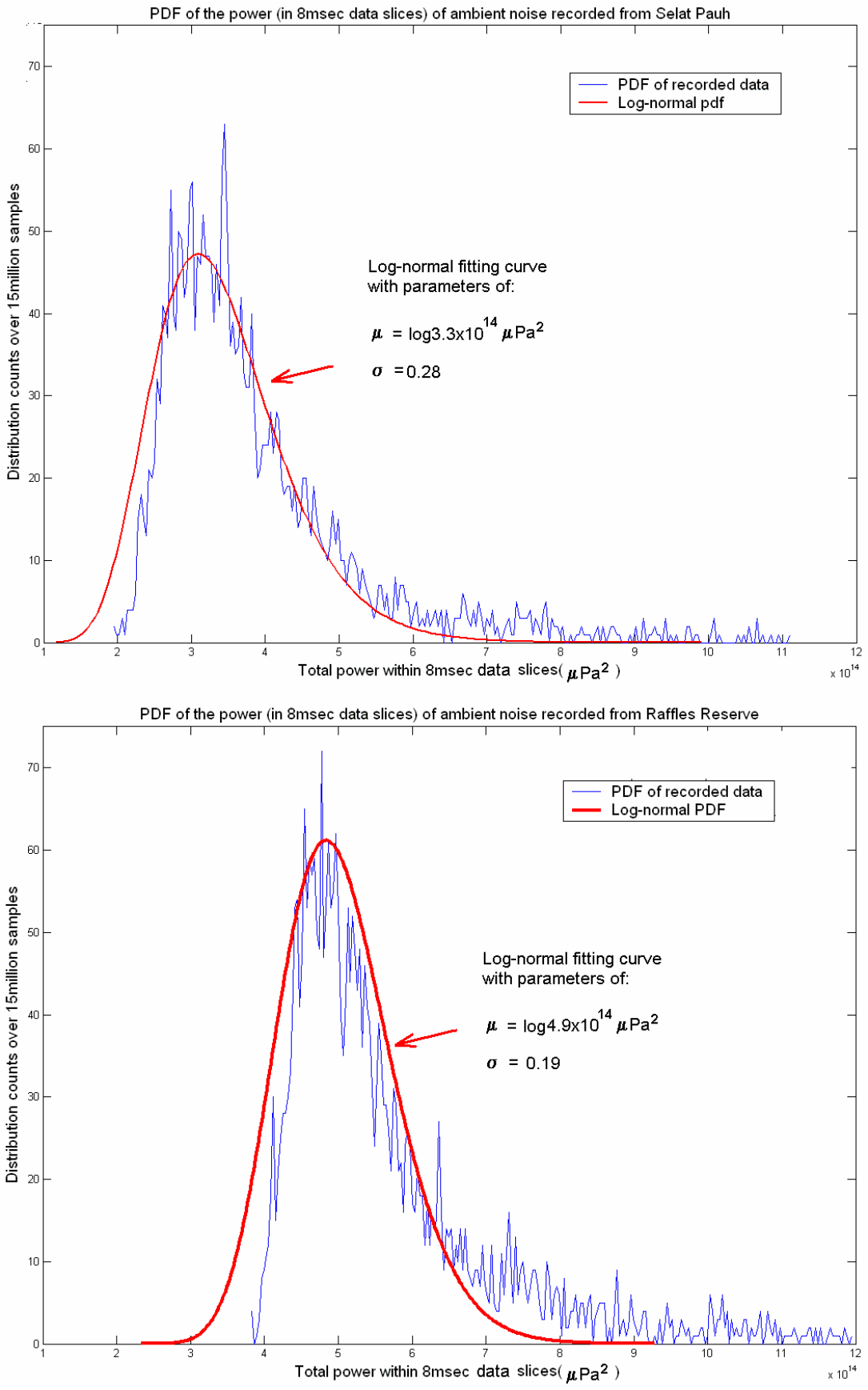


Figure 42: Power distribution density of time series over 20 seconds

7.2 High Frequency Ambient Noise Directivity

Figure 43 shows the azimuth directivity plots (over 360 degree at 1 degree interval) of ambient noise data collected at Selat Pauh and Raffles Reserve. By summing the high frequency source power (800Hz – 200kHz) over all elevation angles for each azimuth direction, the directivity of the ambient noise energy was derived. This was done over all the clicks identified within the data set, including the surface reflected clicks. The difference in total power from each direction was then plotted in dB scale with reference to the lowest energy level observed. The plot shows very significant directivity differences among the sites. It is observed that the ambient noise directivity depends on its relative locations to nearby noise sources (in this case the snapping shrimp clicks) and their density. For example, the array was mounted from a barge, therefore its directivity was dominate by a patch of shrimps living bellow the barge, although there were more noise source at the seabed (see Figure 48 for more explanations).

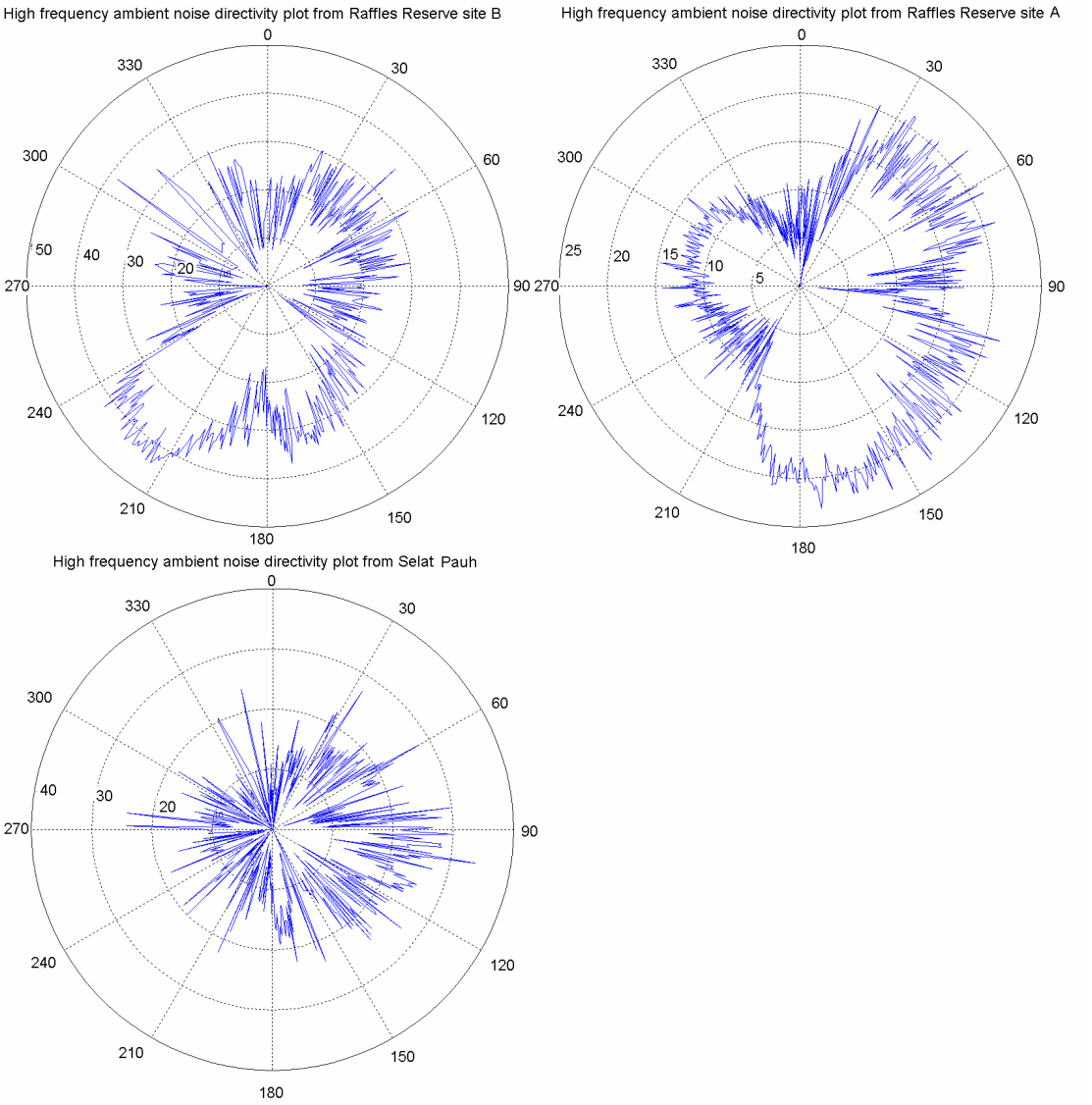


Figure 43: Directivity plots (in dB) of high frequency ambient noise.

7.3 Spatial Distribution of Snap Sources

After the analysis of the acoustic power probability distribution, the spatial distribution density of source snaps was then investigated. The data collected from the four hydrophones was processed to identify individual snaps. The inter-channel time delays of a snap were estimated and used to resolve the direction of the snap. Based on the vertical directions of the snaps, we are able to classified snaps from the surface and seabed. Hence the source location on the seabed or sea surface (assuming the sources are either on seabed or sea surface). The snap occurrence of each look angle (at about 1 degree angular resolution) in three-dimensional space was counted

over the 20 minutes worth of data and the counts were projected in their respective source locations in Cartesian (bottom plane and surface plane) as shown in Figure 44 (Selat Pauh site) and Figure 45 (Raffles Reserve site A). Both sites presented significant spatial distribution patterns, which could be correlated with the habitat preference of the shrimp. The bottom types of the seabed at both areas were known to be a mix of muddy ground and sparse reef patches that could be homes for shrimp colonies.

There is a common observation on all three data sets, which exhibit high-density snaps areas at the surface plane where the barges were located. This suggests that there could be significant population of shrimp at the bottom of the barge. During the experiment at Selat Pauh, the array was deployed from the site of one of the barges anchored at a barge anchorage area. Referring to the spatial distribution map, the array (which is the origin of the plot) was located at the straight edge of high-density snap distribution area, which is highly likely to be the barge. The plot at Selat Pauh shows a second high snap density area from surface, which could be from the shrimp population live at the bottom of another barge anchored nearby. The hypothesis is further supported by the distribution plot from Raffles Reserve site A, where it shows a high snap density area which approximate a rectangular of 27m by 12m, which is about the size of the barge the array was deployed from.

Another observation from these three data sets is that localized snap densities from the bottom are smaller than those from the barge. This could be due to the shrimp at the surface are more active, or it could be simply that the shrimp population at the bottom of the barge are denser. These observations suggest that the system has performed well in mapping the shrimp distribution and secondly, suggest that the snapping shrimp are able to populate the bottom of a moving surface structure.

The result also shows that the shrimp's snap density could differ significantly from the average snap density even within small area with radius of 100m; therefore, researchers should be careful when assuming the snaps density of an area when estimating the ambient noise. For example (refer to

Figure 45), the snaps density on seabed at Raffles Reserve site A could range from 0.0001 to 0.035 snaps/second/meter^2 at different spots within 100-meter radius whilst the overall average snaps density over the entire area is about 0.0006 snaps/second/meter^2. To top it off, the snap density from the sea surface peaks up to about 0.127 snaps/second/meter^2 at the bottom of the barge. This shows a large variation in snap density within a small area.

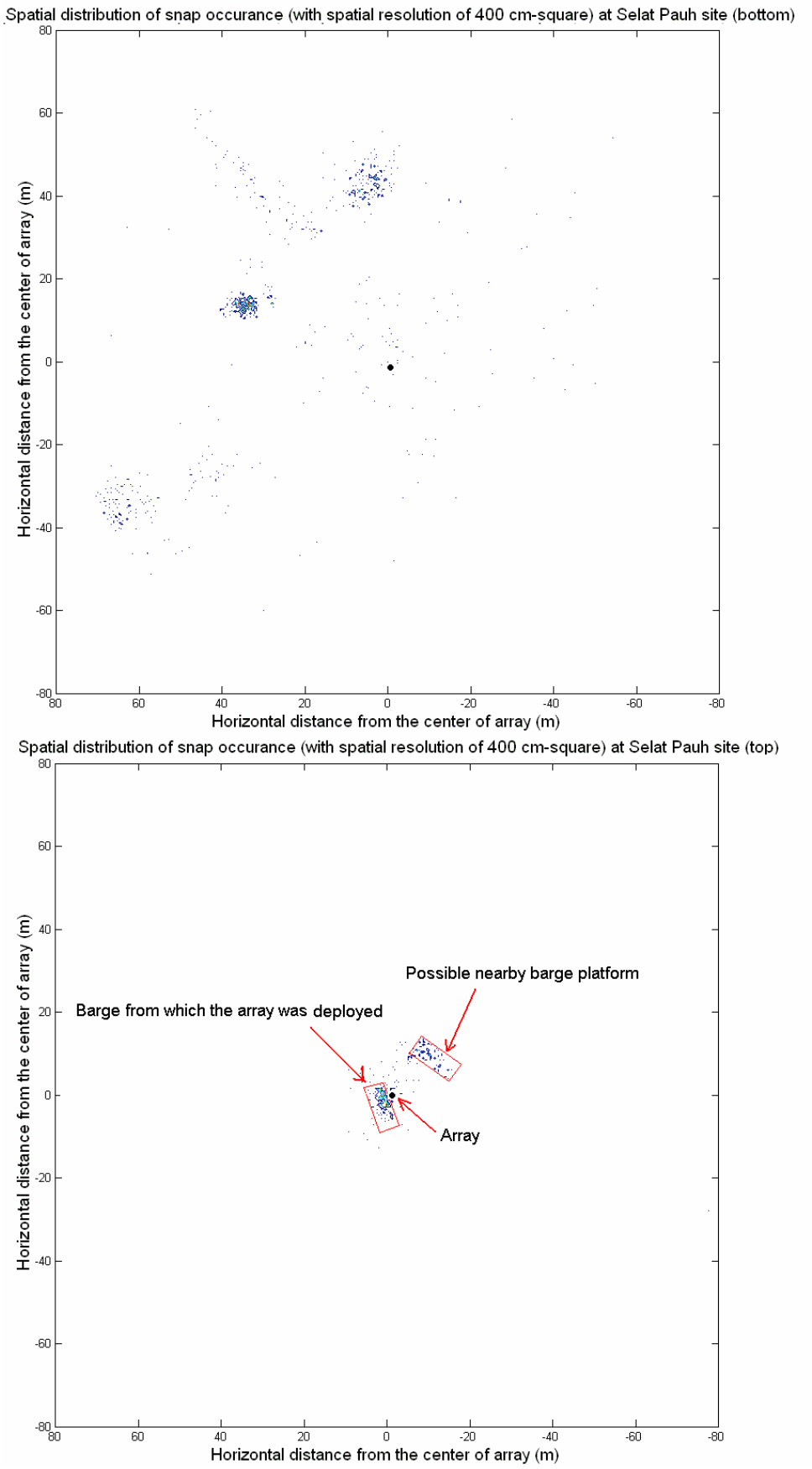


Figure 44: Spatial distribution of snap occurrences at Selat Pauh

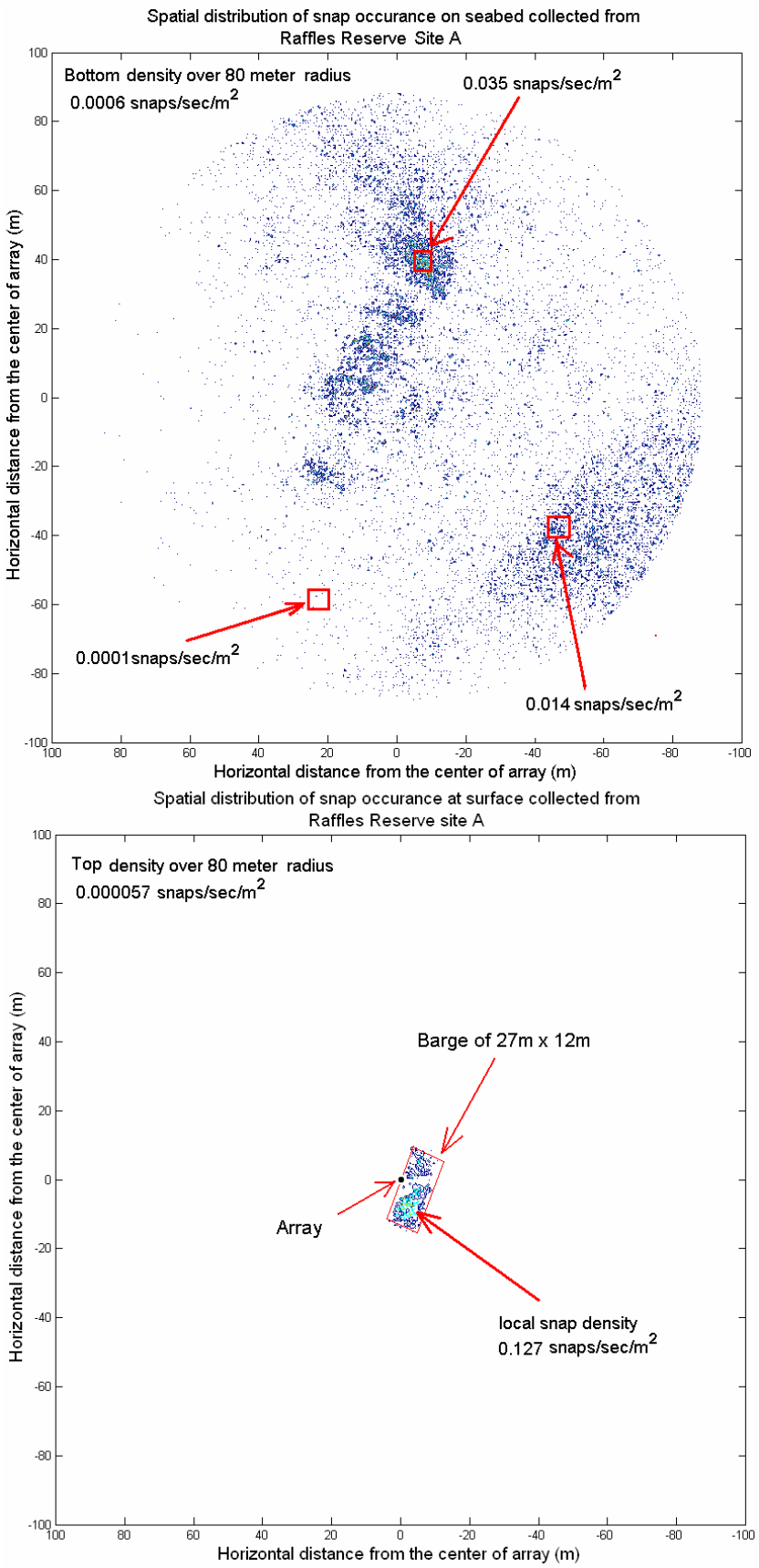


Figure 45: Spatial distribution of snap occurrences at Raffles Reserve site A

7.4 Snapping Shrimp Source Level Estimation

The source level (SL) of each snap was calculated by taking into account the spherical spreading loss of 20logR, where R is the estimated range of each clicks to the centre of the array based on the directions calculated, assuming the shrimps are distributed on seabed. Referring to Figure 46, it was found that the peak-to-peak source levels from seabed were around 175.7 dB re 1μPa @ 1m (standard deviation of 6.3 dB re 1μPa @ 1m), 172.2 dB re 1μPa @ 1m (standard deviation of 4.8dB re 1μPa @ 1m), and 174.2 dB re 1μPa @ 1m (standard deviation of 8.4dB re 1μPa @ 1m) for snaps recorded at Selat Pauh, Raffles Reserve site A and Raffles Reserve site B respectively. On the other hand, source levels from the surface were generally smaller, which were 163.1 dB re 1μPa @ 1m (standard deviation of 12.2 dB re 1μPa), 163.3 dB re 1μPa @ 1m (standard deviation of 7.4 dB re 1μPa @ 1m), and 172.7 dB re 1μPa @ 1m (standard deviation of 7.3 dB re 1μPa @ 1m) respectively. The total numbers of samples in the distributions are different among the distribution plots as the quantity of snaps identified in each site (over a same effective period of time) was different. One possible reason for this is that each site could have different population density and different snaps frequency.

The source level of the snapping shrimp snaps measured were lower than previously reported snapping shrimp click levels [10] in captive environment. This could be due to the variations of bubbles size (hence the acoustic signature of snaps) produced by different species of shrimp, or could be produced by same species but different age of the colony.

It is also observed that generally the surface source level of all three data sets, irrespective of the array location, are lower than the source level from the seabed. Several possible reasons could explain this: for example, some of the sources detected from surface are surface reflections that are naturally smaller in amplitude compared to direct source. Another possible reasons were that the snapping shrimp living on the barge could be different species from the one at the bottom or their physical size could be smaller than

those from bottom due to the poorer living condition. Nevertheless, one of these can be concluded at this time.

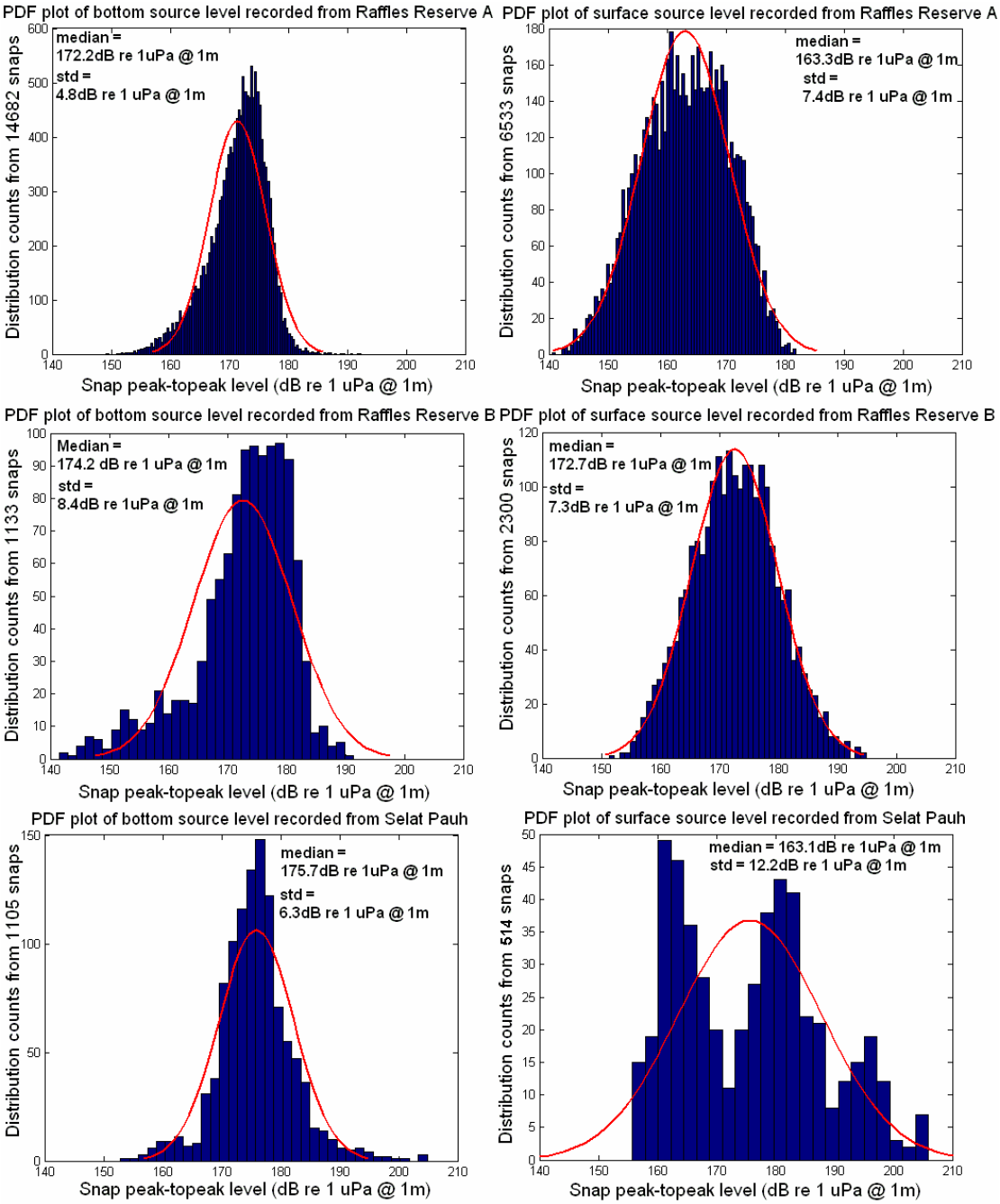


Figure 46: Source level PDF shows median snap power around 172~176 dB re 1 μPa at 1m from bottom and 163~173 dB re 1 μPa at 1m from surface. The red curves are normal fit to the distribution.

Another observation from the data set was that the snap distributions from surface are much better approximates of normal distribution, except the result from Selat Pauh trial where the array could be too close to the surface

to give good range estimation. The huge deviation of bottom source level distribution from normal curve could to be due to the range estimation error caused by uneven bathymetry of the sea bottom that is currently assumed flat by the algorithm. This can be fixed by collecting bathymetry information of the side under investigation and correct the source level accordingly.

After each snaps' source level was estimated, they were plotted on a Cartesian coordinates to form a spatial distribution diagram of source levels. When there were multiple clicks in the same location, the average of these clicks was taken. Figure 47 shows the peak-to-peak snap power of individual snap identified by the algorithm and their locations during the trial at Raffles Reserve site B. The surface distribution also indicates significant snapping shrimp activities at the bottom of the barge from which the array was setup and deployed, with click power of up to 195 dB re 1μPa @ 1m. It is also noted that although the density of the sources on seabed was sparse compare to the bottom of the barge. This could be due to the shrimp are more prefer to the habitats provided by the bottom of the barge than the sea bottom at this area.

The high concentration of source level from the bottom of barge and the sparse distribution from the seabed shows that the high frequency ambient noise directivity at Raffles reserve site B could probably dominated by the biological noise from the surface structure. The upper illustration in Figure 48 shows the relationship of the high frequency ambient noise and the local snapping shrimp distribution in this area. It is clear that the ambient noise level at the direction towards the barge is much larger than other sites.

The location of the array within the water column does affect the ambient noise directivity it receives. For example, when it is near to the bottom (upper plot of Figure 48), the sources on seabed contribute much more to the directivity measured (because they are nearer) than when it is near to surface (see lower plot of Figure 48). In contrast, the source from the bottom of the barge dominates the directivity even if the bottom distribution has more snap occurrences in total.

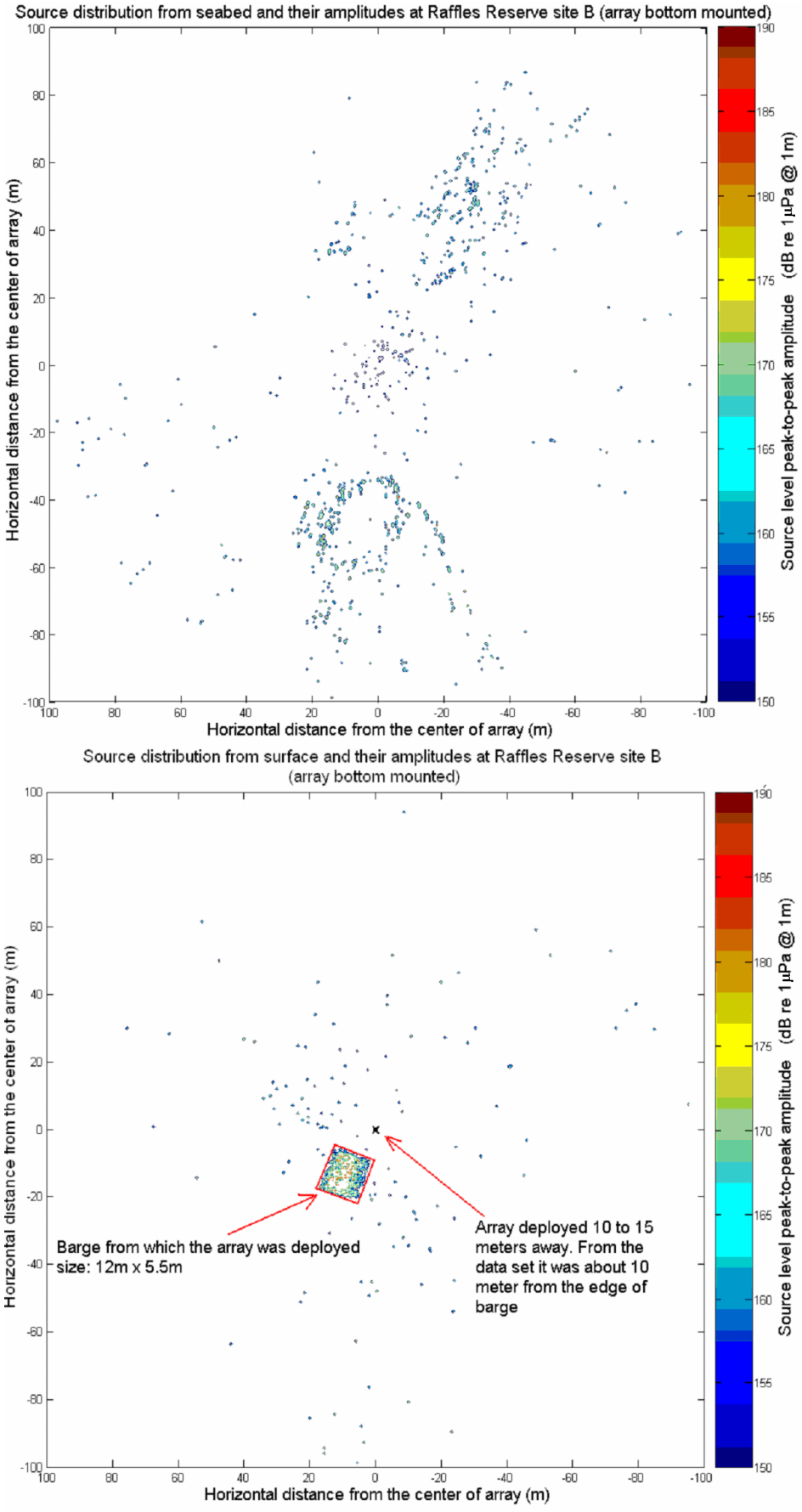


Figure 47: Spatial distribution of mean peak-to-peak source level over 20 minutes

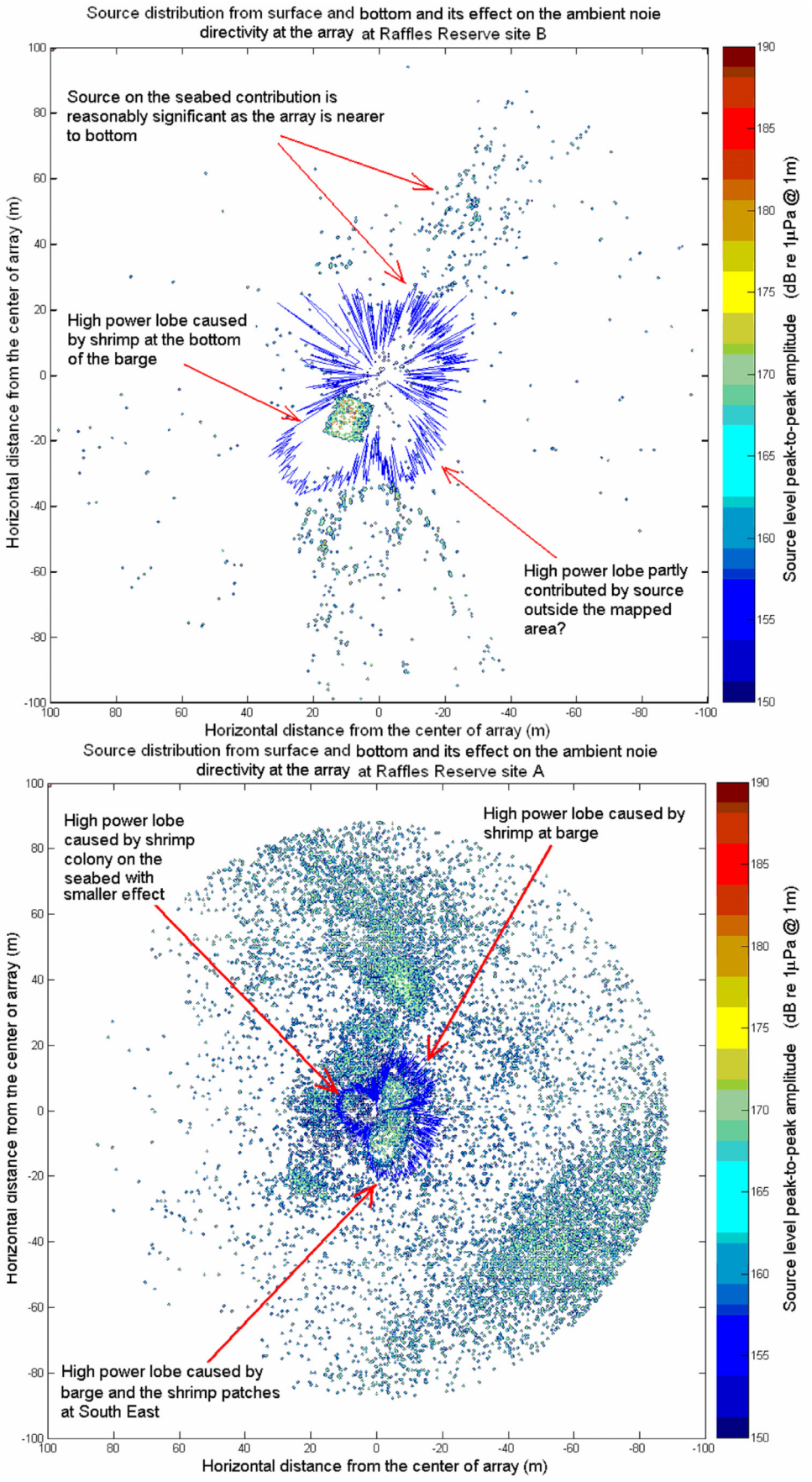


Figure 48: The relationship between high frequency ambient noise directivity at both sites and the nearby snapping shrimp sources.

7.5 Temporal Variation of Snapping Shrimp Clicks

The estimated source levels were then plotted over time to investigate if there was any significant temporal choral. As the acquisition was performed in bursts, with idles in between burst that range from seconds to hours, there were discontinuities between the data sections. Therefore, only the snapping shrimp clicks identified in a same continuous acquisition burst are plotted over time when we look for any pattern of temporal variation. The temporal distribution in Selat Pauh was not plotted during the investigating the temporal coral because its snaps density was too sparse to be plotted over small time windows. Figure 49 shows samples of snaps identified within one acquisition burst (about 28 second per acquisition at site A and about 150 seconds per acquisition at site B). The plot doesn't exhibit any significant clusters (i.e. no particular time with snap density that are higher than others) along the time axis in the plot and hence no significant temporal chorusing were observed.

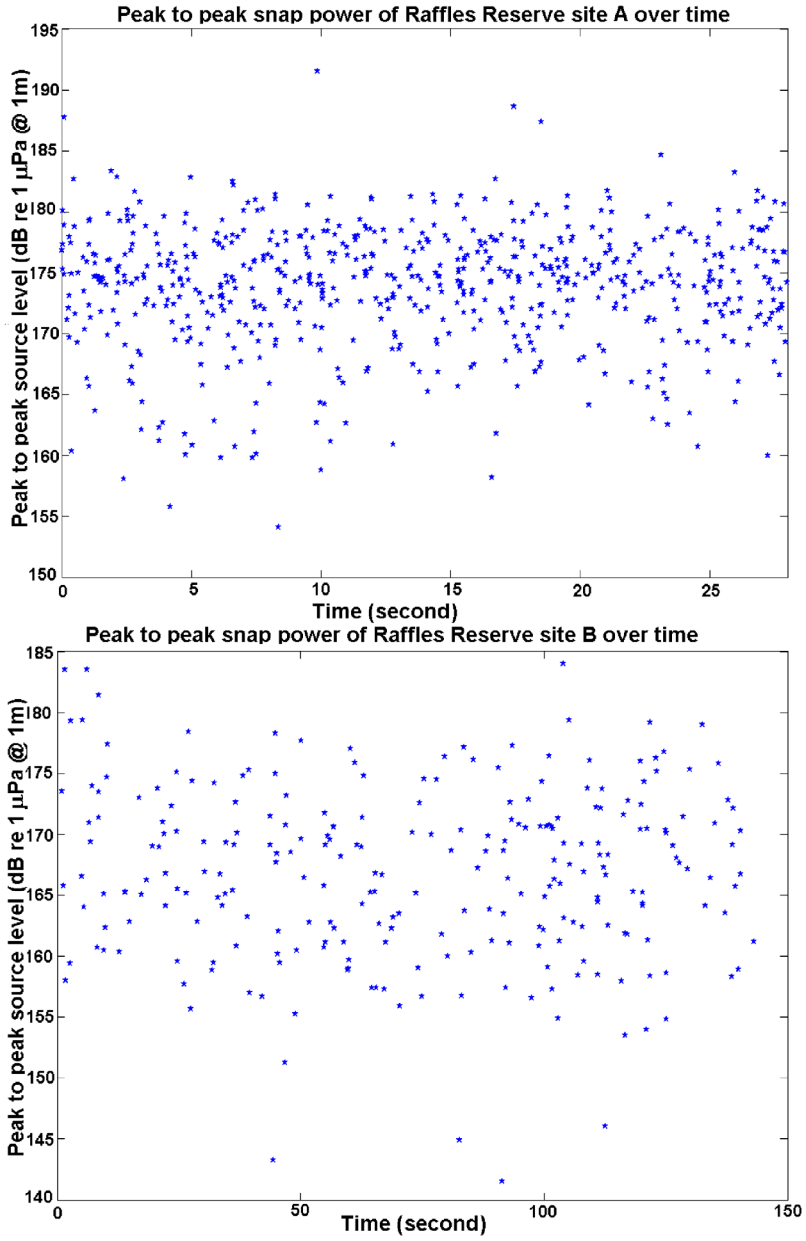


Figure 49: Sample plots of source level at Raffles Reserve sites.

Next was to investigate if the source level varies over time. All the snaps from a same site were divided into sections of 10 seconds and the mean of the source levels within the time windows were calculated. The time stamp of each section was taken as the median of the time of the elements in the group. Finally, the values were plotted over time with error bars set to one standard deviation. As shown in Figure 50, the mean of the source level at both sites seem to be reasonably constant.

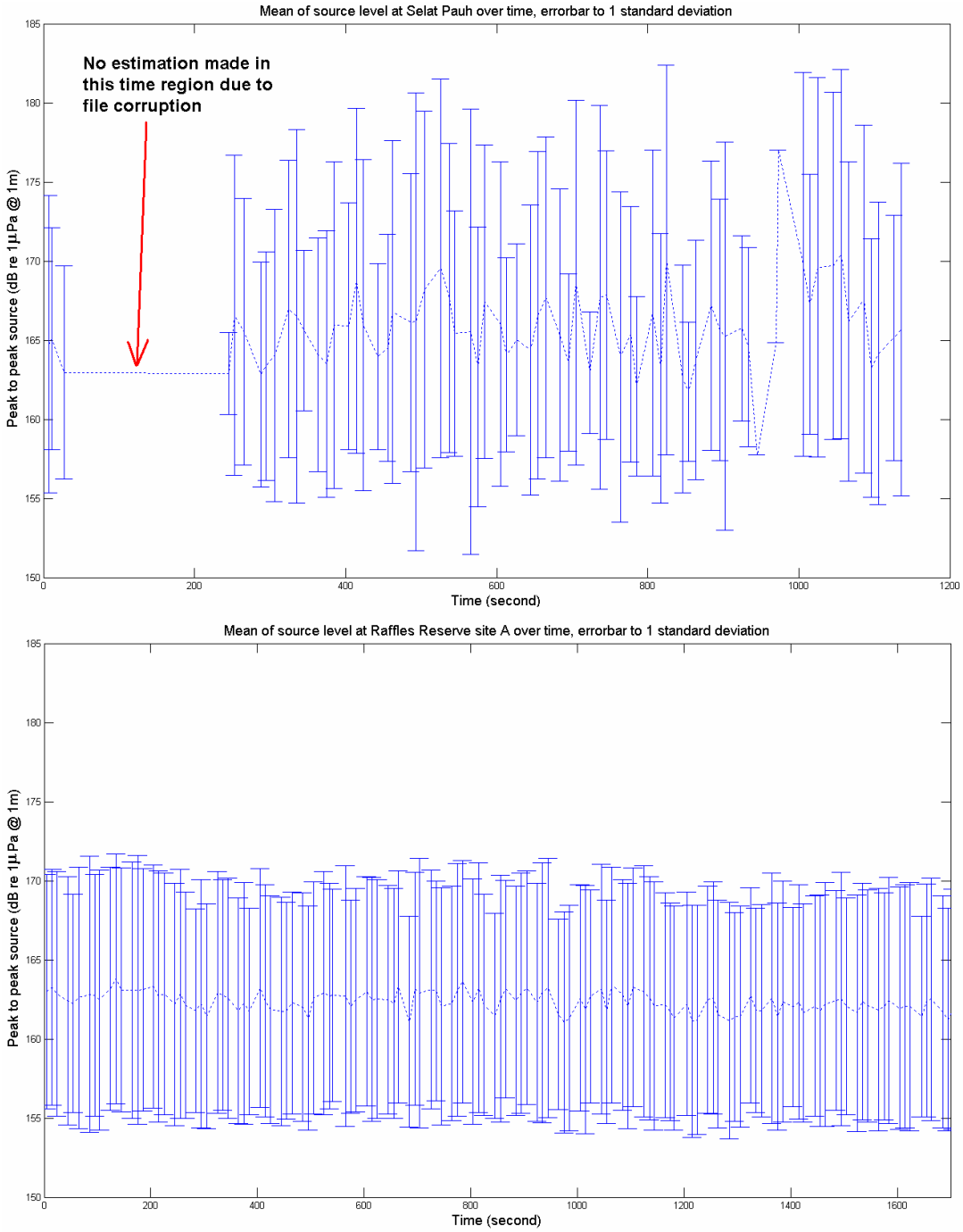


Figure 50: variations of mean of source level over time.

The standard deviations of source levels at Selat Pauh were larger than the ones calculated from Raffles Reserve. This could be understood as the number of snaps recorded from Selat Pauh was smaller and hence the distribution estimates has larger error.

CHAPTER 8
CONCLUSION

The project has developed a portable, easily deployable system for estimating the spatial and temporal distribution of high frequency, broadband acoustic noise generated by snapping shrimp. The compact size of the system and its flexibility allowed it to be rapidly deployed at open waters as well as areas that are remote and confined. The system could be deployed either as bottom mounted system or surface mounted system; and in standalone or in cabled operation. Bottom-mounted configuration is the most favorable setup (although diver support is required to deploy it and the height of array is limited by the support structure, hence it has poorer range resolution when mapping sources from bottom). This is because the array does not subject to positioning and rotational fluctuations when it is bottom mounted. On the other hand, the surface mount configuration allows it to be deployed within 10 minutes even at areas that are geographically restricted and confined by tolerating the higher error level. Choosing the way of mounting is a matter of finding the trade-offs between spatial mapping tolerance, stability, and ease of mounting.

The project has produced several type of high frequency ambient noise study in Singapore for the first time, such as the spatial distribution plot of high frequency source levels (mainly produced by snapping shrimp), the source levels of snapping shrimp snaps in local water, estimations of high frequency ambient noise directivity in local waters, and the investigation of the temporal distribution of the snapping shrimp clicks.

The field results showed significant spatially clustered distributions of noise sources. This provides one potential explanation to the near-lognormal distribution of the ambient noise power in local water [15]. The understanding of ambient noise source location information is important in order to design better acoustics related marine equipment. For systems that see ambient noise as an interference to proper operation (such as conventional sonar, side scan, acoustic modems), understanding the noise will help the designers to

find ways to get around it and increase the signal to noise ratio. On the other hand, for other equipment that utilizes ambient noise (such as ambient noise imaging systems), understanding it will enable the designer to use it with higher efficiency.

Apart from the ambient noise study, the system has also been used to support other experiments that needed to beamform the arrivals of high frequency signals such as the acoustic classification of coral reefs, a study of dolphin bio-sonar, and a study of snapping shrimp acoustics. Two papers describing the HiDAQ system and the results of shrimp distribution study have been published over the last two years [31] [3]. In addition, two conferences presentations were given [5] [4] utilizing HiDAQ.

8.1 Future Work and System Upgrades

The robustness of the system could be further improved for future work. For example, one enhancement could be upgrading the analog board's bandpass filter so that its cut off frequencies (which are currently fixed) can be digitally controlled. This would largely increase the flexibility of the system for other experiments that require different frequency ranges. Another potential enhancement would be to upgrade the current analog signal gain stage from manual switch control to digital control. Currently, an initial acoustic recording has to be done before hand so that the analog gain can be manually set according to the ambient noise condition before the actual experiment. A third enhancement would be to upgrade the battery package to include electronics for charging them internally and avoid having to disassemble and re-assemble the system each time the battery runs out.

This project has developed a system suitable for mapping and recording high frequency source distribution and proved its successful operation. It provides an easy means, for the first time, to carry out an island wide study of the ambient noise soundscape, not only from the aspect of its frequency content but also the aspect of the directivity, spatial and temporal distributions. There are also plans in the future to collaborate with the

Department of Biological Science of the National University of Singapore to further the study of snapping shrimp acoustics and their habitat.

REFERENCES

[1] Urick, R.J., “Principles of Underwater Sound for Engineers”, New York: McGraw Hill. 1967.

[2] John R. Potter, Lim Tze Wei & Mandar Chitre, “High-frequency ambient noise in warm shallow waters”, Sea Surface Sound, UK, 1997

[3] Teong Beng Koay, Eng Teck Tan, Mandar Chitre & John R. Potter, "Estimating the spatial and temporal distribution of snapping shrimp using a portable, broadband 3-dimensional acoustic array", Oceans 2003 Marine Technology and Ocean Science Conference (MTS/IEEE), San Diego, USA, September 22-26, 2003.

[4] Mandar Chitre, Teong Beng Koay & John R. Potter, “Origins of directionality in snapping shrimp sounds and its potential”, Oceans 2003 Marine Technology and Ocean Science Conference (MTS/IEEE), San Diego, USA, September 22-26, 2003

[5] Matthias Hoffmann-Kuhnt, John R. Potter, Adam A. Pack, Teong Beng Koay, Mark H. Deakos, Louis M. Herman & Caroline Durville, “Up close and personal: recording humpback whale song at close ranges (10-50m)”, Oceans 2003 Marine Technology and Ocean Science Conference (MTS/IEEE), San Diego, USA, September 22-26, 2003

[6] Paul J. Seekings, Tze Lyn Loh, John R. Potter & Loke Ming Chou, "Acoustic Backscatter Measurements To Distinguish Between Four Types Of Coral Reef Substrates", Islands and Reefs Seminar, Kuala Lumpur, Aug 15-16, 2003

[7] Lloyd Butler, “Underwater radio communication”, Armateur Radio, April 1987.

[8] Versluis et al., “How Snapping Shrimp Snap: Through Cavitating Bubbles”, *Science* 2000 289: 2114-2117

[9] Potter, J.R., T.W. Lim, and M.A. Chitre. “Ambient noise environments in shallow tropical seas and the implications for acoustic sensing”, in Oceanology International '97. 1997. Singapore: Spearhead Exhibitions Lt

[10] Au W.L. & Banks K., “The acoustics of the snapping shrimp Synalpheus parneomeris in Kaneohe Bay”, pp41-47, J. Acoust. Soc. Am. 103 (1), 1998.

[11] P. Venugopalan, Mandar A. Chitre, Eng Teck Tan, John Potter, Koay Teong Beng, Sheldon B. Ruiz & Soo Pieng Tan, “Ambient Noise Imaging - First Deployments of ROMANIS and Preliminary Data Analysis”, Oceans 2003 Marine Technology and Ocean Science Conference (MTS/IEEE), San Diego, USA, September 22-26, 2003.

[12] Hazen, M.G.; Desharnais, F, “The Eastern Canada Shallow Water Ambient Noise experiment”, OCEANS '97. MTS/IEEE Conference Proceedings, Volume: 1, 6-9 Oct. 1997 Page(s): 471 -476 vol.1

[13] Yang, T.C, Kwang Yoo, “Influence of acoustic environment on the vertical directionality of ambient noise in coastal water”, OCEANS '96. MTS/IEEE. 'Prospects for the 21st Century'. Conference Proceedings, Volume: 1, 23-26 Sept. 1996 Page(s): 351 -357 vol.1

[14] Robinson, E.; McConnell, S., “Sensitivity Of High Frequency Surface-Generated Noise To Sonar And Environmental Parameters”, OCEANS , Volume: 15 , Aug 1983 Page(s): 11 -15

[15] John R. Potter, Lim Tze Wei & Mandar Chitre, "Acoustic Imaging & the Natural Soundscape in Singapore Waters", DSO-NUS Joint R&D Seminar, Singapore, 1997

[16] Ferguson, B.G.; Cleary, J.L., “Estimating the polar distribution of snapping shrimp with a wide aperture array”, Signal Processing and Its Applications, 1999. ISSPA '99. Proceedings of the Fifth International Symposium on , Volume: 2 , 22-25 Aug. 1999 Pages:843 - 846 vol.2.

[17] Miklovic, D.W.; Bird, M.T., ”Snapping shrimp: measuring their natural distribution in space and time with low frequency arrays”, OCEANS, 2001. MTS/IEEE Conference and Exhibition , Volume: 4 , 5-8 Nov. 2001 Pages:2095 - 2099 vol.4

[18] “DAQ PCI-6110/6111 User Manual November 2000 Edition”, Document Part Number 321759C-01, National Instrument, 2000.

[19] “A look at SCSI performance: Fujitsu MAJ3364MC U160-SCSI, SCSI vs. IDE: Performance”. An www.tomshardware.com article.

[20] David Russell and Rainer Bruche, "Online Automatic Discrimination Between Solid and Gaseous Cerebral Microemboli With the First Multifrequency Transcranial Doppler", Stroke, Aug 2002; 33: 1975 - 1980.

[21] Potter J.R., Delory E., Constantin S. & Badiu S., "The thinarray; a lightweight, ultra-thin (8 mm OD) towed array for use from small vessels of opportunity", Underwater Technology 2000, Tokyo, Japan, June 2000

[22] C.D. Motchenbacher and J.A. Connelly, “Low-Noise Electronic System Design”, John Wiley & Sons Inc., USA 1993.

[23] W. Kester, S. Wurcer, and C. Kitchin, “High Impedance Sensors”, Analog Devices, Massechusetts, USA.

[24] Lewis Smith and D.H. Sheingold, “Noise and operational amplifier circuits” Analog Dialogue 25th Anniversary Issue, Analog Devices, Massechusetts, USA, 1999.

[25] Henry W. Ott., “Noise reduction Techniques in Electrical Systems”, Wiley-Interscience Pulblication, USA 1988.

[26] “Ultralow distortion, ultralow noise Op Amp, AD797”, datasheet, Analog Devices, Massechusetts, USA.

[27] “Dual low noise, picoampere bias current JFET input Op. Amp, LT1169”, datasheet, Linear Technologies, California, USA, pp8, 11.

[28] “Very low noise, low distortion, active RC quad universal filter, LTC1562-2”, datasheet, Linear Technologies, California, USA, pp2.

[29] A Michael J. Caruso, “Applications of magnetic sensors for low cost compass systems”, Application Note, Honeywell Inc. SSEC.

[30] John R. Potter, "Ambient Noise Imaging techniques and potential in warm shallow water", Proceedings of the ASW Asia workshop, Singapore, 2003.

[31] Teong Beng Koay, Eng Teck Tan & John R. Potter, “A Portable, Self-contained, 5MSa/s Data Acquisition System for Broadband, High Frequency Acoustic Beamforming”, Oceans 2002 MTS/IEEE Conference and Exhibition, vol 1, Biloxi, 2002.

APPENDICES

A. Related Publications based on HiDAQ

Papers with student as principle author

Teong Beng Koay, Eng Teck Tan, Mandar Chitre & John R. Potter, "Estimating the spatial and temporal distribution of snapping shrimp using a portable, broadband 3-dimensional acoustic array", Oceans 2003 Marine Technology and Ocean Science Conference (MTS/IEEE), San Diego, USA, September 22-26, 2003.

Teong Beng Koay, Eng Teck Tan & John R. Potter, “A Portable, Self-contained, 5MSa/s Data Acquisition System for Broadband, High Frequency Acoustic Beamforming”, Oceans 2002 MTS/IEEE Conference and Exhibition, vol 1, Biloxi, 2002.

Papers with student as co-author

Matthias Hoffmann-Kuhnt, John R. Potter, Adam A. Pack, Teong Beng Koay, Mark H. Deakos, Louis M. Herman & Caroline Durville, “Up close and personal: recording humpback whale song at close ranges (10-50m)”, Oceans 2003 Marine Technology and Ocean Science Conference (MTS/IEEE), San Diego, USA, September 22-26, 2003.

Mandar Chitre, Teong Beng Koay & John R. Potter, “Origins of directionality in snapping shrimp sounds and its potential”, Oceans 2003 Marine Technology and Ocean Science Conference (MTS/IEEE), San Diego, USA, September 22-26, 2003

B. Listings of software

```
C:\old_Ulysis\KoayPersonal\courses\Master_Re...\findClick.m      Page 1
10 February 2004                                              22:32:50

function findClick(fileNameBase,findex,pathName,fileNameResult,compassReading);
%
% findClick(fileNameBase,findex,pathName,fileNameResult,compassReading);
%
% input:
%   fileNameBase: base name of bin file to be processed, the actuall file name is to be↙
 suffixed
%   findex: index of the file
%   pathName:
%   fileNameResult: based name of mat file to be generated, it contains (multiple) file↙
Num
%   compassReading: rad, the reading of the compass
% output:
%   mat file in name of [fileNameResult findex] containing structure fileNum(i) contain↙
ing
%       compassReading: rad,
%       goodcount:
%       badcount:
%       time:           time of the file started (empty at teh moment
%       date:           date of the data recording
%       clickNumber:    total nos click found (unclassified validity
%       clickLocation:  bytes, position of good click's peak from the begining of file
%       clickLocationBad:   bytes, position of good click's peak from the begining of f↙
ile
%       clickSample:    time series of good click, fill downwards the matrix, across↙
% col are diff clicks
%       clickSampleBad:     of bad click
%       clickAmpPkPk:       in quantisation levels, Peak2peak amplitude of good clicks
%       delaysBtwnHyd:      interpolated samples, [d21;d31;d41]
%       direction:          rad, [theta; phi] of good clicks
%       clickEnergy:
%       channelOrientation: change to alter the channel orders [chnl; chn2; chn3; chn4]↙
; %
%       path:   path of the file processed
%       name:   file name of the file proccessed
%       hidaqSpec:  a struct containg the specifications of hidaq; see loadHifdaqSpec
%       setup:      a struct of the experiment setup
%       speedWater: m/s speed of water;
%       clickStatus: bolleen, Status of the click is it good or not?
%       delaysResampleFactor:   resampling factor for the delay calculation
%                               multiply fs to get back original distances
%
%  A snap span over the cut windows would cause multiple snaps detected
%  (detect one peak at one window and another peak at the following)
%
%  Koay Jan 2004

% setup of experiment

% setup equipment
%loadHidaqSpec;
loadHidaqSpec2;
  % Autoload list
% analogBoard = [49.5 55.8 63.6 45.7;...
%        18 18 35.5 11;...
%       1.5 2.3 4.6 1.2];
% hidaq.tetraArmLen = 1.2; %m
% hidaq.gainSetting = 3; % default 64dB gain
% hidaq.voltRangeADC = 5; %V, peak
% hidaq.resADC = (2^12)/2; %quantisation level (one bit for polarity)
```

```
C:\old_Ulysis\KoayPersonal\courses\Master_Re...\findClick.m      Page 2
10 February 2004                                              22:32:50

% hidaq.analogGain = analogBoard(1,hidaq.gainSetting); %dB
% hidaq.analogNoise = analogBoard(2,hidaq.gainSetting); %mVpp
% hidaq.analogNoiseRms = analogBoard(3,hidaq.gainSetting); %mVrms
% hidaq.lowPass = 200e3; %Hz
% hidaq.highPass = 1e3; %Hz
% hidaq.heightArray = 4; %m
% hidaq.snr = 43; %dB, signal to noise ratio, use to generate nosie
% hidaq.sensitivityHydrophone = -212; %dB re 1 uPa @ 1m
%
% % setup source
% speedWater = 1540; %m/s
% setup.heightArray = 4; %m
% setup.waterDepth = 20; %m, water depth

clickmin = 0;
clickmax = 0;

% channel swapping, change 1of want to swap location
chn1 = 1;
chn2 = 2;
chn3 = 3;
chn4 = 4;

% % deciding the delays needed for beamforming
% wtrspeed = speedWater;
% max_delay = 0;
% for sweepPhi = -89:89
%     [d21, d31, d41] = generateDelay(hidaq.tetraArmLen, [], (0:359), sweepPhi);
%     d21 = round(d21*hidaq.fs/wtrspeed);          % deplay in nearest nos sample
%     d31 = round(d31*hidaq.fs/wtrspeed);          % deplay in nearest nos sample
%     d41 = round(d41*hidaq.fs/wtrspeed);          % deplay in nearest nos sample
%     max_delay = max([max_delay ceil(max([max(max(d21,[],2)) max(max(d31,[],2)) max(max↙
x(d41,[],2))])])]);
% end
% clear d21 d31 d41;                    % not needed, clear to save memory

max_delay = 400; %390;

%
% setting for shifting window
%
lclick = 250;               % length of typical snapping shrimp click
windTemplate = 1:lclick;    % window size used to browse through the data
lenWind = length(windTemplate);

cap_wind = (-round(max_delay+lclick*2):round(max_delay+lclick*2));  % capturing window↙
mask
lenCapWin = length(cap_wind);   % length of capturing window

resamplef = 4;                  % resample factor before beamforming/direction search
lenCapWinInterp = lenCapWin*resamplef;  % new capture window length after interp
lClickHalfInterp = round(lclick/2)*resamplef;  % half click length

clickno = 0;

% if user doesn't specified compass corection, read from file
if isempty(compassReading)
    icompName = findstr(pathName,'\');
```

```
    fid = fopen([pathName pathName(icompName(end-1)+1:icompName(end)-1) '.txt'],'r');
    lineCompass = fgetl(fid);
    lineCompass = fgetl(fid);
    iCompass = findstr(lineCompass,',');
    compassReading = str2num(lineCompass(iCompass(1)+1:iCompass(2)))
    fclose(fid);
end


nosf = length(findex); % number of files in the same base name

for count = 1:nosf
    tic

    fileNum(count).compassHeading = compassReading*pi/180;      %
    fileNum(count).goodcount = 0;       % start fresh from each file
    fileNum(count).badcount = 0;        % start fresh from each file
    fileNum(count).time = [];           % to be filled later by hand
    fileNum(count).date = [];
    fileNum(count).clickNumber = []; % to be filled in thte process
    fileNum(count).clickLocation = []; % bytes, position of click's peak from the begin↙
ing of file
    fileNum(count).clickLocationBad = [];
    fileNum(count).clickSample = []; % time series of click, fill downwards the matrix,↙
 across col are diff clicks
    fileNum(count).clickSampleBad = [];
    fileNum(count).clickAmpPkpk = []; % to be filled in the process
    fileNum(count).delaysBtwnHyd = []; % interpolated samples, [d21;d31;d41]
    fileNum(count).direction = []; % rad, [theta; phi]
    fileNum(count).clickEnergy = [];
    fileNum(count).channelOrientation = [chn1; chn2; chn3; chn4]; %
    fileNum(count).path = [];
    fileNum(count).name = [];
    fileNum(count).hidaqSpec = hidaq;
    fileNum(count).setup = setup;
    fileNum(count).speedWater = speedWater;
    fileNum(count).clickStatus = []; % Status of the click is it good or not?
    fileNum(count).delaysResampleFactor = resamplef; % resampling factor for the delay ↙
calculation
                                        %(there for delays has better accuracy),
                                        % multiply fs to get back original distanc↙
es

    clicktime = 0;      % refer to begining of each file

    goodcount = 0;
    badcount = 0;

    fileName = [fileNameBase num2str(findex(count)) '.bin'];
    fprintf('\nWorking on %s\n',[pathName fileName]);
    fileNum(count).path = pathName;
    fileNum(count).name = fileName;

    fileNameDir = dir([pathName fileName]);
    filesize = fileNameDir.bytes; %117118*1024;   % in bytes
    nosWinInBuff = 5000; %(filesize - lheader)/2/4/lclick;      %multiple of lclick, no↙
sWinInBuff*lclick will be nos to read from each channel

    %
    % open file take away the header
    % And skip to desired location
```



```
    %
    sample2skip = 0; %(5+1)*12500+11000 ;
    initial_index = sample2skip;    % if we are skipping any samples, add this to be sy↙
nc w begining of file
    max_skip_block = 500000*30; % block size to skip in 30 sec blocks

    lheader = 267;
    fid = fopen([pathName fileName],'r','ieee-be');
    dummy = fread(fid,lheader,'char');


    bigskip = floor(sample2skip/max_skip_block);
    minorskip = mod(sample2skip,max_skip_block);
    if sample2skip*4*2 < filesize   % when we are not skipping entire file

        if bigskip > 0  % to avoid error when skp too large in one time
            for i=1:bigskip
                %read 2byte and skip 6+(block_in_int16-1)*2*4=throwing away block_in_in↙
t16*2*4
                %(*2 because count in bytes) (*4 because we have 4 channel)
                dummy = fread(fid,1, 'int16', 6 + (max_skip_block-1)*2*4);
            end
        end
        % the remaining samples to skip
        if minorskip > 0
            dummy = fread(fid,1, 'int16', 6 + (minorskip-1)*2*4);
        end


        nosBuff = floor((filesize - lheader)/(2*4*nosWinInBuff*lenWind)); % nos of buff↙
er block
        nosRemSam = mod((filesize - lheader),(2*4*nosWinInBuff*lenWind)); % remaining s↙
amples (smaller than a block)

        for iBuffer = 0:nosBuff - 1
            % load a buffer
            fileLocation = ftell(fid);          % file location of the begining of thi↙
s block
            buffer = fread(fid, [4, lenWind*nosWinInBuff], 'int16');
            % remove mean
            buffer(1,:) = buffer(chn1,:) - mean(buffer(chn1,:));
            buffer(2,:) = buffer(chn2,:) - mean(buffer(chn2,:));
            buffer(3,:) = buffer(chn3,:) - mean(buffer(chn3,:));
            buffer(4,:) = buffer(chn4,:) - mean(buffer(chn4,:));

            [fileNum(count),clickno,goodcount,badcount] = processBuffer(buffer, nosWinI↙
nBuff, fileNum(count),...
                windTemplate, lenWind, cap_wind, lenCapWinInterp, clickno, iBuffer, fil↙
eLocation, lclick,...
                resamplef,lClickHalfInterp,goodcount,badcount,hidaq,speedWater);

        end

        if nosRemSam > 0
            % load a buffer
            fileLocation = ftell(fid);          % file location of the begining of thi↙
s block
            buffer = fread(fid, [4, nosRemSam], 'int16');
            nosRemWin = floor(nosRemSam/(2*4*lenWind));
            % remove mean
            buffer(1,:) = buffer(chn1,:) - mean(buffer(chn1,:));
```

```
            buffer(2,:) = buffer(chn2,:) - mean(buffer(chn2,:));
            buffer(3,:) = buffer(chn3,:) - mean(buffer(chn3,:));
            buffer(4,:) = buffer(chn4,:) - mean(buffer(chn4,:));

            [fileNum(count),clickno,goodcount,badcount] = processBuffer(buffer, nosRemW↙
in, fileNum(count),...
                windTemplate, lenWind, cap_wind, lenCapWinInterp, clickno, iBuffer, fil↙
eLocation, lclick,...
                resamplef,lClickHalfInterp,goodcount,badcount,hidaq,speedWater);

        end

    end

%     ftell(fid)
%     fseek(fid,0,'eof');
%     ftell(fid)

    fclose(fid);        % finished this file
    fileNum(count).goodcount = goodcount;
    fileNum(count).badcount = badcount;

    save([pathName fileNameResult num2str(findex(count))],'fileNum');
    clear fileNum;

    toc
end


function [outStruct,clickno,goodcount,badcount] = processBuffer(buffer, nosWinInBuff, i↙
nStruct, windTemplate, lenWind, cap_wind, lenCapWinInterp, clickno, iBuffer, fileLocati↙
on, lclick, resamplef,lClickHalfInterp,goodcount,badcount,hidaq,speedWater)

% Employing Trial #4, then extract a window of data for each detection, centered at pea↙
k
% threshold with energy: make sure the energy within a windows is more than average of ↙
nominal 'noise'
%threshold = mean(buffer(1,:).^2)*length(buffer)/nosWinInBuff;  % get the mean energy w↙
ithin the window

threshold = sum(buffer(1,:).^2)/length(buffer);  % get the total power within the buffe↙
r

for i = 1:nosWinInBuff - 1

    currentWin = windTemplate + (i-1)*lenWind;

    windenergy = sum(buffer(1,currentWin).^2);  % energy content within window
    if windenergy > threshold*2           % if energy content > mean energy, there is s↙
ignal

        [maxV,I] = max(buffer(1, currentWin).^2);  % find instense with max power withi↙
n window
        I = I + (i-1)*lenWind; % refer the time to begining of buffer

%        fprintf('i=%d\tI=%d\tminwincap=%d\tmaxwincap=%d\n',i,I,min(cap_wind+I),max(cap↙
_wind+I));

        if min(cap_wind+I) > 0 & max(cap_wind+I) < length(buffer)

            % check for transient (criteria: max power should be at least 36 times the ↙
```



```
mean power)
            if maxV > mean(buffer(1, currentWin).^2)*(6)^2  % a snap found

                clickno = clickno + 1;                          % count of detected cli↙
cks

                % *
                inStruct.clickNumber = clickno;

                % interpolate for better accuracy           %% start of interpolated op↙
erations %%
                rbuffer(1,1:lenCapWinInterp) = interp(buffer(1,cap_wind+I),resamplef);
                rbuffer(2,1:lenCapWinInterp) = interp(buffer(2,cap_wind+I),resamplef);
                rbuffer(3,1:lenCapWinInterp) = interp(buffer(3,cap_wind+I),resamplef);
                rbuffer(4,1:lenCapWinInterp) = interp(buffer(4,cap_wind+I),resamplef);
                rbuffer(1:4,1:lClickHalfInterp) = zeros(4,lClickHalfInterp);        % z↙
ero padding both end (by replacing, not appending)
                rbuffer(1:4,end-lClickHalfInterp+1:end) = zeros(4,lClickHalfInterp);

                % xcorr to locate clicks
                click_length = lclick*resamplef;        % click length after interp
                % reference click, take half lclick left and right to obtain template
                template = interp(buffer(1,(round(-lclick/2:lclick/2)+I)),resamplef);
                xcorr_ref = max(xcorr(template));

                [xcorry(1,1), loc(1,1)] = xcorr_find(rbuffer(1,:),lenCapWinInterp,templ↙
ate,click_length,xcorr_ref);
                [xcorry(2,1), loc(2,1)] = xcorr_find(rbuffer(2,:),lenCapWinInterp,templ↙
ate,click_length,xcorr_ref);
                [xcorry(3,1), loc(3,1)] = xcorr_find(rbuffer(3,:),lenCapWinInterp,templ↙
ate,click_length,xcorr_ref);
                [xcorry(4,1), loc(4,1)] = xcorr_find(rbuffer(4,:),lenCapWinInterp,templ↙
ate,click_length,xcorr_ref);

                % * record the location
                % want to record here?

                if min(loc) == -1        % if couldn't find a reasonable click, mark it↙
 as failure
                    badcount = badcount +1;
                    inStruct.clickStatus = 0;
                    inStruct.clickLocationBad = [inStruct.clickLocation (fileLocation +↙
 I*2*4)]; % record position in Byte
                    inStruct.clickSampleBad = [inStruct.clickSample buffer(1,(round(-lc↙
lick/2:lclick/2)+I))'];

                    %                                   table = [clickno; 0; 999; 999; cli↙
cktime; 0; 0; findex(count);clickmin; clickmax];
                else
                    goodcount = goodcount +1;
                    inStruct.clickStatus = 1;

                    inStruct.clickLocation = [inStruct.clickLocation (fileLocation + I*↙
2*4)]; % record position in Byte
                    inStruct.clickSample = [inStruct.clickSample buffer(1,(round(-lclic↙
k/2:lclick/2)+I))'];

                    % average energy within the click
                    fsample = hidaq.fs*resamplef;

                    click_en = (sum((rbuffer(1,loc(1,1):loc(1,1)+click_length).^2)*(1/f↙
```

```
sample)}+ ...
                        sum((rbuffer(2,loc(2,1):loc(2,1)+click_length).^2)*(1/fsample))↙
+ ...
                        sum((rbuffer(3,loc(3,1):loc(3,1)+click_length).^2)*(1/fsample))↙
+ ...
                        sum((rbuffer(4,loc(4,1):loc(4,1)+click_length).^2)*(1/fsample))↙
)/4; % average sum of energy

                    % *
                    inStruct.clickEnergy = [inStruct.clickEnergy click_en];

                    clickmax = (max(rbuffer(1,loc(1,1):loc(1,1)+click_length)) + max(rb↙
uffer(2,loc(2,1):loc(2,1)+click_length)) + ...
                        max(rbuffer(3,loc(3,1):loc(3,1)+click_length)) + max(rbuffer(4,↙
loc(4,1):loc(4,1)+click_length)))/4;

                    clickmin = (min(rbuffer(1,loc(1,1):loc(1,1)+click_length)) + min(rb↙
uffer(2,loc(2,1):loc(2,1)+click_length)) + ...
                        min(rbuffer(3,loc(3,1):loc(3,1)+click_length)) + min(rbuffer(4,↙
loc(4,1):loc(4,1)+click_length)))/4;
                    % *
                    inStruct.clickAmpPkpk = [inStruct.clickAmpPkpk clickmax-clickmin];

                    % click max pow estimation good up to 5%
                    clickmpow = (max((rbuffer(1,loc(1,1):loc(1,1)+click_length).^2))+ .↙
..
                        max((rbuffer(2,loc(2,1):loc(2,1)+click_length).^2))+ ...
                        max((rbuffer(3,loc(3,1):loc(3,1)+click_length).^2))+ ...
                        max((rbuffer(4,loc(4,1):loc(4,1)+click_length).^2)))/4; % avera↙
ge of max power

                    % Calculate delays in terms of traveled distance
                    D = [loc(2,1)-loc(1,1); loc(3,1)-loc(1,1); loc(4,1)-loc(1,1)];
                    %D = [loc(1,1)-loc(1,1); loc(1,1)-loc(2,1); loc(1,1)-loc(3,1); loc(↙
1,1)-loc(4,1)];
                    % * record delays
                    inStruct.delaysBtwnHyd = [inStruct.delaysBtwnHyd D];

                    [thetaE, phiE] = estimateDir(D,speedWater,fsample,hidaq.tetraArmLen↙
);       % cal script to resolve direction
                    % * record directions
                    inStruct.direction = [inStruct.direction [thetaE; phiE]];

                end
            end
        end
    end
end
outStruct = inStruct;
```



```
function sorted = sortSec(toBeSorted)
% sortedSec = sortSec(Sec2Sort)
% sec@Sort is : [clickLocation; phi; theta; srcLevel; ones(1,length(theta)); R]
% sort with ascending time, phi & theta, accumulate the energy in the same
% location

if min(size(toBeSorted)) > 1
    sortedSec = sortrows(toBeSorted', [1])';
    % first run, take away clicks that are too nearby
    % take away duplicates (if their separation is less than width of typical snapping ↙
shrimp snap)
    for i = 2:size(sortedSec,2)
        if (sortedSec(1,i)- sortedSec(1,i-1)) < 250
            sortedSec(5,i-1) = 0;
        end
    end
    % remove any that marked to be removed
    sortedSec = sortrows(sortedSec', [5])';
    i = find(sortedSec(5,:) == 0);
    if isempty(i)
        i = 0;
    end
    sortedSec = sortedSec(:,length(i)+1:end);

    % second run, combine those at the same location
    sortedSec = sortrows(sortedSec', [2 3])';
    for i = 2:size(sortedSec,2)
        if sortedSec(2,i) == sortedSec(2,i-1) & sortedSec(3,i) == sortedSec(3,i-1)
            sortedSec(4,i) = 20*log10(sqrt(10^(sortedSec(4,i)/20)^2 + 10^(sortedSec(4,i↙
-1)/20)^2));
            sortedSec(5,i) = sortedSec(5,i) + sortedSec(5,i-1);
            sortedSec(5,i-1) = 0;
            %            [sortedSec(1,i-1) sortedSec(1,i)]
        end
    end
    % remove any that marked to be removed
    sortedSec = sortrows(sortedSec', [5])';
    i = find(sortedSec(5,:) == 0);
    if isempty(i)
        i = 0;
    end
    sortedSec = sortedSec(:,length(i)+1:end);

    sorted = sortrows(sortedSec', [1])';    % sort it back according to time
else
    sorted = toBeSorted;
end
```

```
function [bottomDistribution, surfaceDistribution, medData, stdData] = drawDistribution↙
(varargin)
% [bottomDistribution surfaceDistribution] = drawDistribution(filename, timeResolution,
%                       orientation, burst_interval,heightArray, analogGain, quantLevel↙
)
%
%   filename: file name of mat file to process, can be a list
%   timeResolution: in what time interval to scan througt he data, energies
%   orientation: bolean, 1 for bottom mounted (tetrahedral is invereted); 0 for surface↙
 mounted tetrahedral as geometry coordinate
%   burst interval: sec, time birst freuency between files in the bin will be added up,↙
 in ms
%   heightArray:    m, height array from seabed
%   analogGain:     dB, electronics gain
%   quantLevel:     V, ADC quantization (VoltageRange/resolution)
%
%
%   R will be differ between top and bottom setup

error(nargchk(1,7,nargin));

filename = varargin{1}
if nargin == 1
    timeResolution = [];
    orientation = 1;
    offset = 0;
    heightArray = [];
    analogGain = [];
    quantLevel = [];
elseif nargin == 2
    timeResolution = varargin{2};
    orientation = 1;
    offset = 0;
    heightArray = [];
    analogGain = [];
    quantLevel = [];
elseif nargin == 3
    timeResolution = varargin{2};
    orientation = varargin{3};
    offset = 0;
    heightArray = [];
    analogGain = [];
    quantLevel = [];
elseif nargin == 4
    timeResolution = varargin{2};
    orientation = varargin{3};
    offset = varargin{4};
    heightArray = [];
    analogGain = [];
    quantLevel = [];
elseif nargin == 5
    timeResolution = varargin{2};
    orientation = varargin{3};
    offset = varargin{4};
    heightArray = varargin{5};
    analogGain = [];
    quantLevel = [];
elseif nargin == 6
    timeResolution = varargin{2};
    orientation = varargin{3};
    offset = varargin{4};
```



```
    heightArray = varargin{5};
    analogGain = varargin{6};
    quantLevel = [];
else
    timeResolution = varargin{2};
    orientation = varargin{3};
    offset = varargin{4};
    heightArray = varargin{5};
    analogGain = varargin{6};
    quantLevel = varargin{7};
end

phi = [];
theta = [];
clickAmpPkpk = [];
clickLocation = [];
R = [];


%loop here for multiples
for loopFile = 1: length(filename)

    load(filename{loopFile});
    % Grap data from structures

    ctrSec = length(fileNum);  %always take the last fileNum(n)

    if ~isempty(fileNum(ctrSec).direction)
        hidaq = fileNum(ctrSec).hidaqSpec;
        setup = fileNum(ctrSec).setup;
        % setup local variable from Spec
        fs = hidaq.fs;
        tetraArmLen = hidaq.tetraArmLen;
        BandWidth = hidaq.lowPass;

        % if user specifies array height, use the one given by the user,
        %   else take from the hidaq spec
        if isempty(heightArray)
            heightArray = hidaq.heightArray;
        end
        if isempty(analogGain)
            analogGain = hidaq.analogGain;
        end
        if isempty(quantLevel)
            quantLevel = hidaq.voltRangeADC/hidaq.resADC;
        end

        depthWaterColumn = setup.waterDepth;

        speedWater = fileNum(ctrSec).speedWater;
        beamWidth = speedWater/BandWidth/tetraArmLen;      % beamwidth in rad
        offset = offset*fs;  % in sample

        % compensate for compass & pan&tile levels

        phi = [phi fileNum(ctrSec).direction(2,:)];
        theta = [theta fileNum(ctrSec).direction(1,:)];
        clickAmpPkpk = [clickAmpPkpk fileNum(ctrSec).clickAmpPkpk];
        clickLocation = [clickLocation fileNum(ctrSec).clickLocation+offset];

    end
```

```
end

% we should limit the range, else the range error will get tooo large
%  and the SL will be not meaningful due to large error
maxRange = 100;                             % interested radius in meter
phiMin = atan(heightArray/maxRange);  % smallest phi for this range

% group surface and bottom according to the sign of phi
group = sortrows([clickLocation;phi;theta;clickAmpPkpk;ones(size(theta))]', [2])';

% mark all the indices with too small of phi (further then
[indice] = find(abs(group(2,:)) < phiMin);
group(5,indice) = 0;
% remove any that marked to be removed
group = sortrows(group', [5])';
i = find(group(5,:) == 0);
if isempty(i)
    i = 0;
end
group = group(:,length(i)+1:end);
fprintf('%d snaps with L too large are removed\n',length(indice));
indice = 0;

indice=find(group(2,:) < 0);
if ~isempty(indice)
    if orientation == 1
        groupSurface = group(:,1:length(indice));
        groupBottom = group(:,length(indice)+1:end);
    else
        groupBottom = group(:,1:length(indice));
        groupSurface = group(:,length(indice)+1:end);
    end
    % estimate range, R, assuming flat seabed
    % estimate range more accurately with bathymetry
    groupSurface(6,:) = abs((depthWaterColumn-heightArray)./sin(groupSurface(2,:)));   ↙
 %m, range to source
    groupBottom(6,:) = abs((heightArray)./sin(groupBottom(2,:)));    %m, range to sourc↙
e
    fprintf('%d snaps from surface\t %d snaps from bottom\n',size(groupSurface,2),size(↙
groupBottom,2));
    indice = 0;
else
    groupSurface = [];
    groupBottom = group;
    if orientation == 1
        groupBottom(6,:) = abs((heightArray)./sin(groupBottom(2,:)));    %m, range to s↙
ource
    else
        groupBottom(6,:) = abs((depthWaterColumn-heightArray)./sin(groupBottom(2,:))); ↙
   %m, range to source
    end
end

% calculate source level and correct for spreading loss
if ~isempty(groupSurface)
    groupSurface(4,:) = 20*log10(groupSurface(4,:)*quantLevel)-analogGain-...
        hidaq.sensitivityHydrophone + 20*log10(groupSurface(6,:));
end
groupBottom(4,:) = 20*log10(groupBottom(4,:)*quantLevel)-analogGain-...
    hidaq.sensitivityHydrophone + 20*log10(groupBottom(6,:));
```



```
% round theta & phi to its beamwidth pi/180 (0.5deg)
radp5 = 0.5*pi/180;
theta = round(theta/radp5)*radp5;
phi = round(phi/radp5)*radp5;


% %------------ generate map over time----------

if ~isempty(timeResolution)
    % set time resolution
    %    timeResolution = 300; %ms
    timeResolution = timeResolution/1000;
    timeResSample = fs*timeResolution;  %equivalent time wind in samples
    timeWin = 1:fs*timeResolution;  %equivalent time wind in samples

    %Sort with time

    % put the data in the time Window
    % in struct, struct index is time, in multiples of timeResolution

    % for surface
    if ~isempty(groupSurface)
        sortedSurface = sortrows(groupSurface',[1])';
        sortedSurface = splitNSort(sortedSurface,timeResSample);
    end

    % for bottom
    sortedBottom = sortrows(groupBottom',[1])';
    sortedBottom = splitNSort(sortedBottom,timeResSample);
else
    % for surface
    if ~isempty(groupSurface)
        timeBin = 1;
        sortedSec = sortSec(groupSurface);
        sortedSurface(timeBin).phi = sortedSec(2,:);
        sortedSurface(timeBin).theta = sortedSec(3,:);
        sortedSurface(timeBin).srcLevel = sortedSec(4,:);
        sortedSurface(timeBin).snapDensity = sortedSec(5,:);
        sortedSurface(timeBin).R = sortedSec(6,:);
        sortedSurface(timeBin).clickLocation = sortedSec(1,:);
    end

    % for bottom
    timeBin = 1;
    sortedSec = sortSec(groupBottom);
    sortedBottom(timeBin).phi = sortedSec(2,:);
    sortedBottom(timeBin).theta = sortedSec(3,:);
    sortedBottom(timeBin).srcLevel = sortedSec(4,:);
    sortedBottom(timeBin).snapDensity = sortedSec(5,:);
    sortedBottom(timeBin).R = sortedSec(6,:);
    sortedBottom(timeBin).clickLocation = sortedSec(1,:);

end


% % plot distribution
%
% dataPlot = sortedBottom;
%
% % for loop is usefull when the data is sectioned into time windows
% for i = 1:length(dataPlot)
```

```
%
%     [X,Y,Z,Zden,xx,yy] = scaleLinXY(dataPlot(i), 500, maxRange, heightArray);
% %    [X,Y,Z,x,y] = scaleLogXY(dataPlot(i),heightArray,beamWidth,phiMin);
% %    [X,Y,Z,xx,yy] = scaleSparse(dataPlot(i),1);
%
%     plotDist(X,Y,Z,Zden,xx,yy);
%     pause(2);
% end

% dataPlot = sortedSurface;
% for i = 1:length(dataPlot)
%     [x,y,z] = sph2cart(dataPlot(i).theta,dataPlot(i).phi,dataPlot(i).R);
%     [X,Y] = meshgrid(x,y);
%     [X,Y,Z] = griddata(x,y,dataPlot(i).srcLevel,X,Y);
%     figure(1);
%     hold on;
%     contour(X,Y,Z);
%     %axis([-100 100 -100 100]);
%     pause(2);
% end

% output
if nargout == 2
    bottomDistribution = sortedBottom;
    if ~isempty(groupSurface)
        surfaceDistribution = sortedSurface;
    else
        surfaceDistribution = [];
    end
elseif nargout == 4
    bottomDistribution = sortedBottom;
    if ~isempty(groupSurface)
        surfaceDistribution = sortedSurface;
    else
        surfaceDistribution = [];
    end
    medData = median(sortedBottom.srclevel);       % mean square
    stdData = std(sortedBottom.srclevel);
end


% figure;
% pd = sortedBottom.srcLevel;
% maxpow = max(pd);
% powbin = maxpow/100;
% hist(pd,0:powbin:maxpow);
% meanpow = median(pd)        % mean square
% stdpow = std(pd)



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       Support routines
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function splitedSorted = splitNSort(toBeSplit,timeResSample)
% split into time resolution given, add all the snaps power and occurance in it
%   timeResSample: time reoultion, in samples
%
%
```



```
timeBin = 1;  % should be initialise earlier
timeGate = timeBin*timeResSample;
indTimeTag = 1;

lenClickLocation = length(toBeSplit(1,:));
for indTime = 2:lenClickLocation
    if  toBeSplit(1,indTime) > timeGate | indTime >= lenClickLocation
        % consolidate cl;ick in the timeWind
        toBeSplit = sortSec(toBeSplit(:,indTimeTag:indTime-1)');
        %             [clickLocation(indTimeTag:indTime-1); phi(indTimeTag:indTime-1);.↙
..
        %                     theta(indTimeTag:indTime-1); srcLevel(indTimeTag:indTime-↙
1);...
        %                     ones(1,length(indTimeTag:indTime-1)); R(indTimeTag:indTim↙
e-1)]);
        % store them
        sortedData(timeBin).phi = toBeSplit(2,:);
        sortedData(timeBin).theta = toBeSplit(3,:);
        sortedData(timeBin).srcLevel = toBeSplit(4,:);
        sortedData(timeBin).snapDensity = toBeSplit(5,:);
        sortedData(timeBin).R = toBeSplit(6,:);
        sortedData(timeBin).clickLocation = toBeSplit(1,:);
        % update count
        timeBin = timeBin +1;
        timeGate = timeBin*timeResSample;
        indTimeTag = indTime;
    end
end
splitedSorted = sortedData;
```

```
function [pdOut,Zout,X,Y,axisRange] = plotDist(varargin)
% [pdOut,Zout,X,Y,axisRange] = plotDist(X,Y,Z,Zden,xx,yy,axisLimit)
% input:
%   X,Y: vector to scale the 2 dim plot, X(j), Y(i)
%   Z:     dB re 1uPa @ 1m, two dimensional matrix of the distribution Z(i,j)
%   Zden:  snaps, snap density within the area
%   xx,yy:  original x,y coordinate of the source
%   axisLimit: [lowerLimit, upperLimit] of colormap, willautogenerate to the nearest po↙
ssible
%  OR
%
% [pdOut,Zout,X,Y,axisRange] = plotDist(dataStruct, resolution, maxRange, heightArray, ↙
axisLimit)
%
% input:
%    data: struct containing
%        srcLevel (dB), theta (rad), phi (rad), snapDensity (snaps)
%   npixels: number of pixels for the plot
%   maxRadius: (m), max radius of area interested
%   heigthArray: (m), array height, not used at the moment
%   axisLimit: [lowerLimit, upperLimit] of colormap, willautogenerate to the nearest po↙
ssible
% ouput:
%    X,Y: vector of the coordinate, m
%    Zout: distribution of source rms within the area of pixel
%
%   Koay Jan 2004

if nargin == 4
    dataStruct = varargin{1};
    resolution = varargin{2};
    maxRange = varargin{3};
    heightArray = varargin{4};
    if isempty(heightArray)
        [X,Y,Z,Zden,xx,yy,pd] = scaleLinXY(dataStruct, resolution, maxRange);
    else
        [X,Y,Z,Zden,xx,yy,pd] = scaleLinXY(dataStruct, resolution, maxRange, heightArra↙
y);
    end
    upperLimit = [];
    lowerLimit = [];
    axisLimit = [];
elseif nargin == 5
    dataStruct = varargin{1};
    resolution = varargin{2};
    maxRange = varargin{3};
    heightArray = varargin{4};
    if isempty(heightArray)
        [X,Y,Z,Zden,xx,yy,pd] = scaleLinXY(dataStruct, resolution, maxRange);
    else
        [X,Y,Z,Zden,xx,yy,pd] = scaleLinXY(dataStruct, resolution, maxRange, heightArra↙
y);
    end
    axisLimit = varargin{5};
    if length(axisLimit) == 1
        lowerLimit = axisLimit(1,1);
        upperLimit = [];
    elseif length(axisLimit) == 2
        lowerLimit = axisLimit(1,1);
        upperLimit = axisLimit(1,2);
    end
```



```
elseif nargin == 6
    X = varargin{1};
    Y = varargin{2};
    Z = varargin{3};
    Zden = varargin{4};
    xx = varargin{5};
    yy = varargin{6};
    upperLimit = [];
    lowerLimit = [];
    axisLimit = [];
    pd = [];
elseif nargin == 7
    X = varargin{1};
    Y = varargin{2};
    Z = varargin{3};
    Zden = varargin{4};
    xx = varargin{5};
    yy = varargin{6};
    axisLimit = varargin{7};
    if length(axisLimit) == 1
        lowerLimit = axisLimit(1,1);
        upperLimit = [];
    elseif length(axisLimit) == 2
        lowerLimit = axisLimit(1,1);
        upperLimit = axisLimit(1,2);
    end
    pd = [];
else
    error('Wrong number of input argument');
end


% avoid divide by 0
[i,j] = find(Zden == 0);
for icount = 1:length(i)
    Zden(i(icount),j(icount)) = 1;
end
%    snapPow = 20*log10(sqrt(((10.^(dist./20)).^2)./distDensity));  % rms of power
Z = 20*log10(sqrt((10.^(Z./10))./Zden));  % rms of power
% limit the colormap
if isempty(axisLimit)
    [i,j,s]=find(Z);
    minDist = min(s)-1;
    [i,j] = find(Z <= 0);
else
    minDist = lowerLimit - 1;
    [i,j] = find(Z <= minDist);
end

for icount = 1:length(i)
    Z(i(icount),j(icount)) = minDist;
end
if ~isempty(upperLimit)
    % patch a value at far end, they are inaccurate anyway
    Z(length(Z)-1:end,length(Z)-1:end) = upperLimit;
end

minZ = min(min(Z,[],2));
maxZ = max(max(Z,[],2));
fprintf('Zmin = %.1f\t Zmax = %.1f\n',minZ,maxZ);
% plot contour
```

```
contour(X,Y,Z);
%pcolor(X,Y,Z);


if nargout == 1
    pdOut = pd;
elseif nargout == 2
    pdOut = pd;
    Zout = Z;
elseif nargout == 4
    pdOut = pd;
    Zout = Z;
    X = X;
    Y = Y;
elseif nargout == 5
    pdOut = pd;
    Zout = Z;
    X = X;
    Y = Y;
    axisRange = [minZ, maxZ];
end
```



```
function [X,Y,Z,xx,yy] = scaleLogXY(data,heightArray,beamWidth,phiMin)
% [X,Y,Z] plotLogXY(data,heightArray,beamWidth,phiMin)
% generate X(j),Y(i),and Z(i,j) for 3D and contour plot according to range resolution (↙
m)
%   from spherical coordinate, and source info
% input:
%   data: struct containing
%       srcLevel (dB), theta (rad), phi (rad), snapDensity (snaps)
%   heigthArray: (m)
%   beamWidth: beamWidth of the beam interested (rad)
%   phiMin: (rad), any data with phi magnitude less than this will be
%           ignored, range error too large
% ouput:
%   X,Y: vector of the coordinate, m
%   Z: surface of the data (dB)
%
% koay Jan 2004

% convert to cartesian
[x,y,z] = sph2cart(data.theta,data.phi,data.R);

% round to give matrix indece
xP = round((180/pi*atan2(x,heightArray)+90)/(beamWidth*180/pi))+1;
yP = round((180/pi*atan2(y,heightArray)+90)/(beamWidth*180/pi))+1;
% arrange z
z = zeros(length(-90:(beamWidth*180/pi):90));
for icount = 1:length(x)
    %z(yP(icount),xP(icount)) = 20*log10(sqrt(10.^(data.srcLevel(icount)./10)./data.sna↙
pDensity(icount)));
    z(yP(icount),xP(icount)) = 20*log10(sqrt((z(yP(icount),xP(icount))^2+10.^(data.srcL↙
evel(icount)./10))./data.snapDensity(icount)));
end
% scale x & y
icount = 1:length(z);
icount = ((icount - 1)*(beamWidth*180/pi)-90)*pi/180;
xL = heightArray*tan(icount);
yL = heightArray*tan(icount);
% remove indices with too large range error
lim = (length(-90:(beamWidth*180/pi):90)-length(-(90-ceil(phiMin*180/pi)):(beamWidth*18↙
0/pi):(90-ceil(phiMin*180/pi))))/2;

% set output
if nargout == 3
    Z = z(lim:end-lim+1,lim:end-lim+1);
    X = xL(lim:end-lim+1);
    Y = yL(lim:end-lim+1);
elseif nargout == 5
    Z = z(lim:end-lim+1,lim:end-lim+1);
    X = xL(lim:end-lim+1);
    Y = yL(lim:end-lim+1);
    xx = x;
    yy = y;
end
```

```
function [amp,pos] = xcorr_find(data,ldata,template,ltemplate, reflvl)
% [Yxcor, pos] = XCORR_FIND(data,ldata,template,ltemplate, reflvl)
%   relvl: the best xcorr level (auto corr of snap)
%   template: a template of the snap
%   data to search through
%   finds and returns the position of template with some checking



%[xcorry, xloc] = max(xcorr(data,template));  % find click
%xloc = xloc - ldata;
[y,i] = xcorr(data,template);
[xcorry,I] = max(y);
xloc = i(I);
% if xloc < 0
%       plot_wind;
%       pause;
% end


if xloc < 0          % if lcick is out of range, ignore it and find another click
    %data(1,1:round(ltemplate/2)) = zeros(1,round(ltemplate/2));
    data(1,1:ltemplate) = zeros(1,ltemplate);
    [y,i] = xcorr(data,template);
    [xcorry,I] = max(y);
    xloc = i(I);
elseif (xloc + max(ltemplate) > ldata -1 )        % if the clicks extense more than the↙
 window, it couldn't be possible a same click
    data(1,xloc:ldata-1) = zeros(1,ldata-xloc);
    [y,i] = xcorr(data,template);
    [xcorry,I] = max(y);
    xloc = i(I);
end

while (xcorry > 2*reflvl) %xloc > 0 & ((xcorry > 2*reflvl) | (xcorry < 0.5*reflvl)) %  ↙
find click that is not unreasonably large (xcorry > 1.5*reflvl) & (xcorry > 0.75*reflvl↙
)
    %plot(xloc:xloc+ltemplate-1)
    data(1,(xloc:xloc+ltemplate-1)) = zeros(1,ltemplate); % data(1,(xloc:xloc+ltemplate↙
-1)) = zeros(1,ltemplate);
    [y,i] = xcorr(data,template);
    [xcorry,I] = max(y);
    xloc = i(I);
end

if xloc < 0          % if lcick is out of range, ignore it and find another click
    %data(1,1:round(ltemplate/2)) = zeros(1,round(ltemplate/2));
    data(1,1:ltemplate) = zeros(1,ltemplate);
    [y,i] = xcorr(data,template);
    [xcorry,I] = max(y);
    xloc = i(I);
elseif (xloc + max(ltemplate) > ldata -1 )        % if the clicks extense more than the↙
 window, it couldn't be possible a same click
    data(1,xloc:ldata-1) = zeros(1,ldata-xloc);
    [y,i] = xcorr(data,template);
    [xcorry,I] = max(y);
    xloc = i(I);
end

if xcorry > 0.5*reflvl & xloc > 0 & floor(xloc + ltemplate/2) < ldata  % if click is re↙
asonable large enough
```



```
    amp = xcorry;            % return value
    pos = xloc;
else                         % else return error
    amp = -1;
    pos = -1;
end
```

```
function [X,Y,Zpow,Zden,xx,yy,pdOut,timeLoc] = scaleLinXY(varargin)
% [X,Y,Zpow,Zden,xx,yy,pdOut,timeLoc] = scaleLinXY(data, npixels, maxRadius, arrayDistF↙
romPlane)
% input:
%   data: struct containing
%       srcLevel (dB), theta (rad), phi (rad), snapDensity (snaps)
%   npixels: number of pixels for the plot
%   maxRadius: (m), max radius of area interested
%   arrayDistFromPlane: (m),
%
% ouput:
%   X,Y: vector of the coordinate, m
%   Zpow: distribution of total power in pixel (dB), sqrt(sum of squares of peak-peak p↙
ressure)
%         note: empty Zpow is defined as 0, therefore will have bug if signal is 1uPa
%   Zden: distribution of density (snaps)
%   xx,yy: original coordinates (m)


% npixels=500;
% maxrange=60;
% data=botDist;
% height=4;

error(nargchk(3,4,nargin));

data = varargin{1};
npixels = varargin{2};
maxRadius = varargin{3};

dist = zeros(npixels,npixels);
distDensity = zeros(npixels,npixels);
maxrange = maxRadius*2;

if nargin == 4 & ~isempty(varargin{4})
    % use the array height given by the user
    [xc,yc]=pol2cart(data.theta,varargin{4}./tan(data.phi) );
    % rescale the range
    data.srcLevel = data.srcLevel - 20*log10(data.R) + 20*log10(abs(varargin{4}./sin(da↙
ta.phi)));
else
    [xc,yc,zc]=sph2cart(data.theta,data.phi,data.R);
end
% scale x,y
qxc=floor(((xc/maxrange)*npixels))+ floor(npixels/2)+1;
qyc=floor(((yc/maxrange)*npixels)) + floor(npixels/2)+1;

% fill the matrix, in linear space
for xcount=1:length(qyc)
    if sqrt(xc(xcount)^2 + yc(xcount)^2) < maxRadius
        dist(qyc(xcount),qxc(xcount))= sqrt((dist(qyc(xcount),qxc(xcount)).^2 + 10^(dat↙
a.srcLevel(xcount)./20).^2));
    %    distDensity(qyc(xcount),qxc(xcount)) = data.snapDensity(xcount) + 1;
        distDensity(qyc(xcount),qxc(xcount)) = data.snapDensity(xcount) + distDensity(q↙
yc(xcount),qxc(xcount));
    end
end
% scale the axis
x = 1:length(dist);
Xloc = (x-1-floor(npixels/2)).*maxrange/npixels;
Yloc = (x-1-floor(npixels/2)).*maxrange/npixels;
```



```
% convert all non zeros in dist to log space, Note: shall not thread zeros as in log sp↙
ace
[i,j] = find(dist);
for icount = 1:length(i)
    dist(i(icount),j(icount)) = 20*log10(dist(i(icount),j(icount)));
end

[i,j,s] = find(dist);
[m,n] = size(dist);
dist = sparse(i,j,s,m,n);

[i,j,s] = find(distDensity);
[m,n] = size(distDensity);
distDensity = sparse(i,j,s,m,n);
clear s;

pd = data.srcLevel;
meanpow = median(pd)        % mean square
stdpow = std(pd)

% output
if nargout == 4
    Zpow = dist;
    Zden = distDensity;
    X = Xloc;
    Y = Yloc;
elseif nargout == 6
    Zpow = dist;
    Zden = distDensity;
    X = Xloc;
    Y = Yloc;
    xx = xc;
    yy = yc;
elseif nargout == 7
    Zpow = dist;
    Zden = distDensity;
    X = Xloc;
    Y = Yloc;
    xx = xc;
    yy = yc;
    pdOut = pd;
elseif nargout == 8
    Zpow = dist;
    Zden = distDensity;
    X = Xloc;
    Y = Yloc;
    xx = xc;
    yy = yc;
    pdOut = pd;
    timeLoc = data.clickLocation;
elseif nargout == 0

    [i,j] = find(distDensity == 0);
    for icount = 1:length(i)
        distDensity(i(icount),j(icount)) = 1;
    end

%    snapPow = 20*log10(sqrt(((10.^(dist./20)).^2)./distDensity));  % rms of power
    snapPow = 20*log10(sqrt((10.^(dist./10))./distDensity));  % rms of power

    % limit the colormap
```

```
    [i,j,s]=find(snapPow);
    minDist = min(s)-1;
    [i,j] = find(snapPow <= 0);
    for icount = 1:length(i)
        snapPow(i(icount),j(icount)) = minDist;
    end

    %contour(Xloc,Yloc,dist), colormap(hot); shading interp;
    pcolor(Xloc,Yloc,snapPow), colormap('hot'); shading interp;
    axis('equal');

%    keyboard;
end


% function plotdist(data, npixels, maxrange)
% % npixels=500;
% % maxrange=60;
% %
% % data=botDist;
%
% height=4;
% dist=zeros(npixels,npixels);
% %maxradius=hieght/tan(max(data.phi))
%
% [xc,yc]=pol2cart(data.theta,4./tan(data.phi) );
%
% qxc=floor(((xc/maxrange)*npixels))+ floor(npixels/2)+1;
% qyc=floor(((yc/maxrange)*npixels)) + floor(npixels/2)+1;
%
%
%
% for x=1:length(qyc)
%
%
%       dist(qyc(x),qxc(x) )= sqrt(  (dist(qyc(x),qxc(x) ).^2+ data.srcLevel(x).^2)/2);
%
%
% end
```

```
function [d21, d31, d41, R] = generateDelay(varargin)
% function [d21, d31, d41, R] = generateDelay(L, angular_resolution,theta,phi,speedWate↙
r,heightArray)
%
%   Caluculates the delays between the sensors at
%       tip of tetrahedran spaced at length L
%       All angular values in rad
%
%   d21, d31, d41 are in length unit (m)
%   incoming wave definition (direction)
%   will generate 360 sweep if obmitted
%   theta = azimuth angle (anti-clockwise +ve), default 0:ang_res:2*pi-ang)res
%   phi = tilt angle (up +ve), default pi/4 (45deg)
%   looking from the top of hydrophone
%   ang_res = angular resolution, default pi/180 (ideg)
%
%   It gives a matrix of theta's by phi's (actual elements depends on the resolution)
%   where values of the indices is the delay of the respective (theta,phi) direction
%
%   uses: calculateDelay
%   by Koay for Hifreq Amb Noise

error(nargchk(0,6,nargin));

if nargin == 0
    % find_delay()
    L =1.2;
    ang_res = 1;
    theta = (0:ang_res:2*pi-ang_res);
    phi = pi/4*ones(size(theta)); %0;
    speedWater = 1540;
    heightArray = 4;
elseif nargin == 1
    % fins_delay(L)
    L = varargin{1};
    ang_res = 1;
    theta = (0:ang_res:2*pi-ang_res);
    phi = pi/4*ones(size(theta)); %;
    speedWater = 1540;
    heightArray = 4;
elseif nargin == 2
    % fins_delay(L, angular_resolution)
    L = varargin{1};
    ang_res = varargin{2};
    theta = (0:ang_res:2*pi-ang_res);
    phi = pi/4*ones(size(theta)); %0;
    speedWater = 1540;
    heightArray = 4;
elseif nargin == 3
    % fins_delay(L, angular_resolution, theta)
    L = varargin{1};
    ang_res = varargin{2};
    theta = vargin{3};
    phi = pi/4*ones(size(theta)); %0;
    speedWater = 1540;
    heightArray = 4;
elseif nargin == 4
    % fins_delay(L, angular_resolution,theta,phi)
    L = varargin{1};
    ang_res = varargin{2};
    theta = varargin{3};
```



```
    phi = varargin{4};
    if length(phi) == 1
        phi=phi*ones(size(theta));
    elseif length(phi) ~= length(theta)
        error('theta & phi must have same size');
    end
    speedWater = 1540;
    heightArray = 4;
elseif nargin == 5
    % fins_delay(L, angular_resolution,theta,phi,speedWater)
    L = varargin{1};
    ang_res = varargin{2};
    theta = varargin{3};
    phi = varargin{4};
    if length(phi) == 1
        phi=phi*ones(size(theta));
    elseif length(phi) ~= length(theta)
        error('theta & phi must have same size');
    end
    speedWater = varargin{5};
    heightArray = 4;
else
    % fins_delay(L, angular_resolution,theta,phi,speedWater)
    L = varargin{1};
    ang_res = varargin{2};
    theta = varargin{3};
    phi = varargin{4};
    if length(phi) == 1
        phi=phi*ones(size(theta));
    elseif length(phi) ~= length(theta)
        error('theta & phi must have same size');
    end
    speedWater = varargin{5};
    heightArray = varargin{6};
end

if isempty(L)
    L =1.2;
end
if isempty(ang_res)
    ang_res = 1;
end
if isempty(theta)
    theta = (0:ang_res:360-ang_res);
end
if isempty(phi)
    phi = 45*ones(size(theta)); %;
end
if isempty(speedWater)
    speedWater = 1540; %m/s
end

for i = 1:length(theta)
    [D21(i),D31(i),D41(i)] = calculateDelay(theta(i),phi(i),L);
end

if length(heightArray) == 1 | length(heightArray)~=length(D21)
    fprintf('R is autogenerated\n');
    R = heightArray./sin(phi);
end
```

```
d21 = D21;
d31 = D31;
d41 = D41;

%     phi = tempphi;
%     d21 = -r.*cos30*cos(phi).*cos(theta) + r.*cos(phi).*sin(theta).*(sin30+1);
%     d31 = r.*cos30*cos(phi).*cos(theta) + r.*cos(phi).*sin(theta).*(sin30+1);
%     d41 = r.*sin(phi).*sqrt(2) + r.*cos(phi).*sin(theta);
```



```
function [X,Y,Z,xx,yy] = scaleSparse(data,plotType)

[x,y,z] = sph2cart(data.theta,data.phi,data.R);

% process 2 types
if plotType == 1
    [Xloc,Yloc] = meshgrid(x,y);
    [Xloc,Yloc,z] = griddata(x,y,20*log10(sqrt(10.^(data.srcLevel/10)./data.snapDensity↙
)),Xloc,Yloc,'cubic');
else
    % x and y has the same size and the data is listed as x(j),y(i),Z(i,j)
    z = zeros(length(x),length(x)); % generate a zero matrix
    for icount = 1:length(x)
        z(icount,icount) = 20*log10(sqrt(10.^(data.srcLevel(icount)/10)./data.snapDensi↙
ty(icount)));
    end
%     %   use sparse
%     [i,j,s]=find(z);     % sometimes too sparse to generate contour
%     z = sparse(i,j,s);
    Xloc = x;
    Yloc = y;
end

% set output
if nargout == 3
    Z = z;
    X = Xloc;
    Y = Yloc;
elseif nargout == 5
    Z = z;
    X = Xloc;
    Y = Yloc;
    xx = x;
    yy = y;
end
```

```
function [dataN, snr, dataC] = simulateSource(varargin)
% [dataNoise, snr, dataClean] = simulateSource(D, R, fname, hidaq, template, speedWater↙
)
%
%   input:
%   D: sec,   inter channels delay with respective direction [d21;d31;d41]
%       Default:array height 4 meter off bottom, sources are at a ring radius
%               30 meter away (about phi = 7.6deg)
%
%   R: m, range of the source, default is calculated from hydaq = 4m above seabed
%   *** Range must be specified if D is given, else, asummed unity level across all are↙
a
%
%   hidaq: structure with spec of hidaq, interested in  heightArray, tetraArmLen, fs, s↙
nr
%       Default:hidaq.heightArray = 4; %m
%               hidaq.tetraArmLen = 1.2; %m
%               hidaq.fs = 500e3; %Hz
%               hidaq.voltRangeADC = 5; %V, peak
%               hidaq.analogNoise = 32; %mV
%               hidaq.snr = 43; %dB, signal to noise ratio, use to generate nosie
%
%   speeWater:  m/s, speed of water default 1540
%   tempalte:   trasncient to duplicate, default snap.mat
%   fbame:      filename of the output, without extension, default 'RSYC'
%
%   output:
%   dataNoise:  generated signals with noise
%   snr:        the SNR used
%   dataClean:  data with only delays
%
%   Note source Levels are all 190re 1uPa @ 1m
%
% Koay Jan 2004


error(nargchk(0,5,nargin));
error(nargchk(0,4,nargout));

% Prepare inputs
% ------------------------------
if nargin == 0
    D = [];
    R = [];
    hidaq = [];
    fname = [];
    template = [];
    speedWater = [];
elseif nargin == 1
    D = varargin{1};
    fname = [];
    R = [];
    hidaq = [];
    template = [];
    speedWater = [];
elseif nargin == 2
    D = varargin{1};
    R = varargin{2};
    fname = [];
    hidaq = [];
    template = [];
```



```
    speedWater = [];
elseif nargin == 3
    D = varargin{1};
    R = varargin{2};
    fname = varargin{3};
    hidaq = [];
    template = [];
    speedWater = [];
elseif nargin == 4
    D = varargin{1};
    R = varargin{2};
    fname = varargin{3};
    hidaq = varargin{4};
    template = [];
    speedWater = [];
elseif nargin == 5
    D = varargin{1};
    R = varargin{2};
    fname = varargin{3};
    hidaq = varargin{4};
    template = vararfin{5};
    speedWater = [];
else
    D = varargin{1};
    R = varargin{2};
    fname = varargin{3};
    hidaq = varargin{4};
    template = vararfin{5};
    speedWater = vararfin{6};
end

if isempty(hidaq)
    % Array parameter
%     hidaq.heightArray = 4; %m
%     hidaq.tetraArmLen = 1.2; %m
%     hidaq.fs = 500e3; %Hz
%     hidaq.voltRangeADC = 5; %V, peak
%     hidaq.analogNoise = 32; %mV
%     hidaq.snr = 43; %dB, signal to noise ratio, use to generate nosie
    loadHidaqSpec;
end

if isempty(speedWater)
    speedWater = 1540; %m/s
end
% simulate source
if isempty(D)
    sL = 30; %m, source radious
    theta = pi/180*(0:1:360-1); % over 2pi
    phi = atan(hidaq.heightArray/L); % find the angle
    R = hidaq.heightArray/sin(phi); %

    [d21,d31,d41] = generateDelay(hidaq.tetraArmLen,[],theta,phi); % calculate delays f↙
or all snaps
    D = [d21;d31;d41]; %m, delay matrix
end
if isempty(fname)
    fname = 'RSYC';
end
if isempty(template)
    load snap; % load the snapping shrimp snap
```

```
else
    structTemplate = load(template);
    nameOfTemplate = fieldnames(S);
    snap =nameOfTemplate;
    clear structTemplate nameOfTemplate;
end
% ------------------------------------
% end prepare inputs
%


% Start of main process

D = round(D/speedWater*(hidaq.fs)); % in sample unit, delay matrix, rounded


lenSnap = length(snap);
snap = snap/(max(snap)-min(snap)); % normalise snap to 1 pk pk

%snap = [snap; snap; snap; snap]; % duplicate 4 channels

data = [];
% in sample unit, Length of each small windows to be combined, each has a click
blockLen = 4*ceil(hidaq.tetraArmLen/speedWater*hidaq.fs);
refPos = ceil(0.25*blockLen):ceil(0.25*blockLen)+lenSnap-1;

for loopCount = 1:length(D)
    block = zeros(4,blockLen);   % emtpy matriex
    % simulate a source distribution
    block(1,refPos) = snap;
    block(2,refPos+D(1,loopCount)) = snap;
    block(3,refPos+D(2,loopCount)) = snap;
    block(4,refPos+D(3,loopCount)) = snap;
    % amplitude correction
    if isempty(R)
        amp = 190 + hidaq.analogGain + hidaq.sensitivityHydrophone;
    else
        amp = 190 + hidaq.analogGain + hidaq.sensitivityHydrophone - 20*log10(R(loopCou↙
nt));
    end
    block = round(block.*(10^(amp/20)/hidaq.voltRangeADC*hidaq.resADC));
    data = [data block]; % merge into data
end

% add noise
noise = 10^((-hidaq.snr-6)/20)*max(data(1,:))*rand(1,length(data)); % denegrate one
noise = [noise;noise;noise;noise];
% data(1,:) = 1024*data(1,:)/max(data(1,:));
% data(2,:) = 1024*data(2,:)/max(data(2,:));
% data(3,:) = 1024*data(3,:)/max(data(3,:));
% data(4,:) = 1024*data(4,:)/max(data(4,:));
dataNoise = data + noise;

% Output
if nargout == 3
    dataN = dataNoise;
    dataC = data;
    snr = hidaq.snr;
elseif nargout == 2
    dataN = dataNoise;
    snr = hidaq.snr;
```



```
elseif nargout == 1
    dataN = dataNoise;
else
    fid = fopen([fname '0.bin'],'w','ieee-be');
    fwrite(fid,ones(1,267),'char');
    fwrite(fid,data,'int16');
    fclose(fid);

    fid = fopen([fname 'N0.bin'],'w','ieee-be');
    fwrite(fid,ones(1,267),'char');
    fwrite(fid,dataNoise,'int16');
    fclose(fid);

%     fid = fopen([fname '.bin'],'r','ieee-be');
%     a = fread(fid,267,'char');
%     data2=fread(fid,[4,inf],'int16');
%     fclose(fid);

    if 0
        figure
        subplot(4,1,1); plot(data(1,:));
        subplot(4,1,2); plot(data(2,:));
        subplot(4,1,3); plot(data(3,:));
        subplot(4,1,4); plot(data(4,:));
    end


end
```