A Wireless PDA-based Acoustics Measurement Platform

Petros Alexandridis[1], Nicolas-Alexander Tatlas[1], Panos Hatziantoniou[1], John Mourjopoulos[1]

[1] Audio and Acoustic Technology Group, Electrical Engineering and Computer Technology Department, University of Patras, 26500, Greece

mourjop@wcl.ee.upatras.gr

ABSTRACT

The proposed platform allows acoustic measurements via a flexible, portable system, based on commercially available hardware, such as a personal digital assistant (PDA) equipped with a wireless adapter and a digital audio capture card as well as a personal computer (PC) interconnected with off-the-shelf wireless networking hardware. Using this hardware, three software applications were implemented: (i) a device driver that handles the communication of the digital audio capture card with the PDA, (ii) a PDA application that realizes the WiFi connection with the personal computer, also incorporating a recording function that captures data and presents the user with their analysis, and (iii) the personal computer application that initiates the playback sequence as dictated by the connected PDA. The system can assist the fast measurement of large spaces. Room Impulse Response (RIR) measurement tests were conducted in a laboratory room, in order to evaluate the effectiveness and functionality of the measurement system.

1. INTRODUCTION

In room response measurement situations where the distance between the input audio device (microphone) and the recording system is often long, practical difficulties arise (e.g. heavy equipment has to be moved for each position). There is a growing need to introduce portable, flexible systems, to automate processes during the measurement and provide the engineer with the proper visual representation of the process as it advances. The introduction of wireless technology (802.11) together with the use of Personal Digital Assistants (PDAs) can assist such tasks.

In this work, the PDA is used as a “smart terminal” and it is connected with an input audio device (microphone through an A/D converter and the appropriate digital capture card). The PDA is at the same time wirelessly connected to the system which is responsible for reproducing the excitation signal and serves as the main control unit. Hence, the audio engineer has full control of the measurement procedure at the desired position and can validate each recording through the data displayed on the PDA.

Although most modern PDAs have the ability to record analog audio through pre-installed microphones, the recording quality is not acceptable for audio applications; furthermore, the current PDA functions are not developed to the level that engineering applications can be fully exploited. A significant contribution of this work is to analyze and explain the features of such platforms and to describe a driver structure for audio applications which can be combined with external audio cards supporting sampling rates of at least 44 kHz and 16 bit quantization.

The key aspects of the relevant technologies are briefly described in Section 2, while a detailed description of the implementation of the proposed system is presented in Section 3. Various aspects of the implementation were tested and validated such as the accuracy of the recorded data, the speed of processing and visual representation, as well as the communication with the playback unit. The power management of the portable device and the processing load was also an important consideration for the implementation. Some measurement experiments and results are presented in Section 4 and conclusions are discussed in Section 5.

2. OVERVIEW

A series of technologies were utilized during the development of the measurement system. These include the use of modern programming frameworks such as the .NET [1] and the DirectX [2] frameworks, audio analysis functions and windows network sockets [3].

The technologies were used to meet the following design goals:

- Simplicity on the user interface.
- Accuracy in the recorded audio signals.
- Effective use of the available hardware.

The simplicity on the user interface was achieved by carefully selecting and organizing the controls that the audio engineer has access to. Everything that is immediately related to the recording process is put directly in the front screen while other options of configuration importance are located on a separate menu. Figure 2 shows an example user interface for the application on the PDA. The .NET framework simplifies the process of building functional user interfaces since it provides a set of graphical tools and functions that can be easily imported into custom projects. Moreover, programs written with the use of the .NET framework can be executed on both portable (PDAs) and desktop computers while offering the exact same user experience and functionality.

The effective use of the hardware was achieved by carefully managing the available resources on each platform. Most PDAs are equipped with embedded processors that have significantly less computational power than modern desktop computers. In order to preserve a responsive user interface and to minimize the time required to compute large processes (such as the FFT of a captured signal) various methods were used. These methods include the use of threads that are executed in parallel to the thread that handles the user interface, the efficient use of the available memory and the optimization of the functions that handle large buffers with audio data.

Finally, various tests were performed during the development of the system in order to verify the accuracy of the recorded audio data and impulse response measurements were performed in a controlled environment such as a professional laboratory room.

3. IMPLEMENTATION

The acoustic response measurement system was implemented, using the following components:

(a) Hardware

- A digital audio capture card with a compact flash form factor [4].

- A typical x86 compatible computer equipped with a sound card and a WiFi adapter. This computer functions as the server of the wireless network and accepts requests from wireless clients. Furthermore, the clients dictate the performance of specific actions such as the initialization of the playback procedure and the adjustment of the playback volume. The x86 computer is also referred as the “base station” or the “server”.

- A personal digital assistant (PDA) equipped with a built-in WiFi adapter, a compact flash socket and an optional memory expansion slot. The PDA is the client that connects to the server, utilizes the digital audio capture card, controls the measurement procedure wirelessly and presents the audio engineer with the analyses of the captured data.

- An SPDI/F microphone or an analog microphone with an analog to digital converter.

(b) Software

Two software applications and a device driver were implemented in order to interconnect the above hardware:

- The application on the PDA that:

  - Controls the digital capture card with the use of the device driver.

  - Transfers the audio data from the digital capture card’s driver to permanent storage.

  - Analyses the audio signal and updates the user interface with plots on the time and frequency domain.

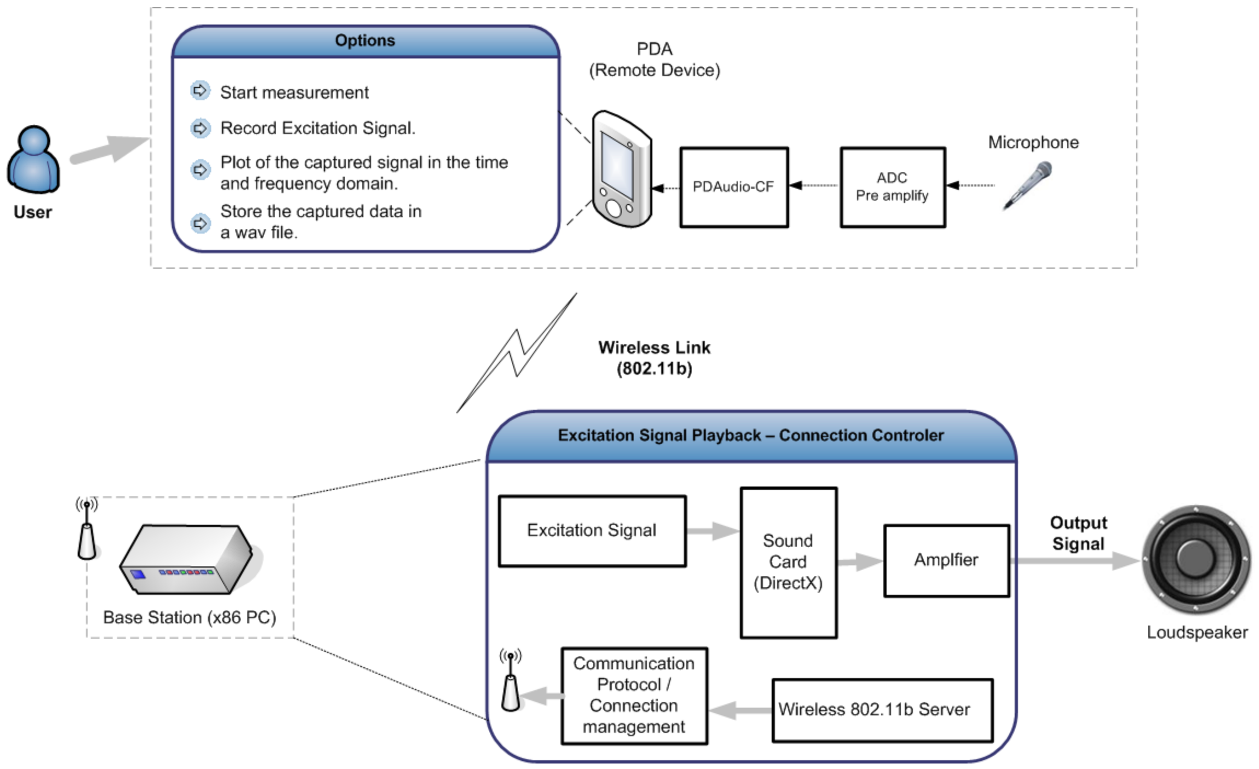
  - Connects to the base station with the use of the WiFi adapter and controls the measurement procedure.

- The application on the base station that:

  - Utilizes the WiFi connection and waits for the PDA to connect.

  - Initializes the sound card with the use of DirectX technology and loads the excitation signal that will be reproduced when dictated by the PDA.

- Finally, the digital capture card driver that transfers the captured audio data from the internal device memory to a specified driver memory.



Options
Start measurement
Record Excitation Signal.
Plot of the captured signal in the time and frequency domain.
Store the captured data in a wav file.
PDA (Remote Device)
PDAudio-CF
ADC Pre amplify
Microphone
User
Wireless Link (802.11b)
Excitation Signal Playback – Connection Controler
Excitation Signal
Sound Card (DirectX)
Amplfier
Output Signal
Loudspeaker
Base Station (x86 PC)
Communication Protocol / Connection management
Wireless 802.11b Server

Figure 1: The complete setup of the system

The topology of the system is represented in Figure 1. All the hardware components are presented together with the software designed specifically for them. Generally, the complete system functions as a data source and data sink: The base station's audio signal fed to the loudspeaker through an external amplifier is the output of this system, while the microphone connected to the digital capture card of the PDA provides input to the system. However, the key difference from other such setups is the fact that the input and the output are two separate entities wirelessly interconnected. The audio engineer can easily relocate the input (PDA with the connected microphone) to any desired location and be guaranteed that the system will function in the exact same way in each new position (limited only by the Wireless Network maximum range). This opens new possibilities in measurements at locations that were previously inaccessible due to the large distance or the topology of the site [5].

The core components of the system are described and briefly analyzed in the following sections.

3.1. PDA Application

The user application on the PDA ("Sweep Capture") was designed to have a minimal set of controls such as a volume scrollbar, a menu with various options, the Play/Record buttons and a real-time sound volume meter. While simple, this user interface does not limit the functionality and the usability of the application. In contrast, various complex operations are executed in background threads without the user interaction.

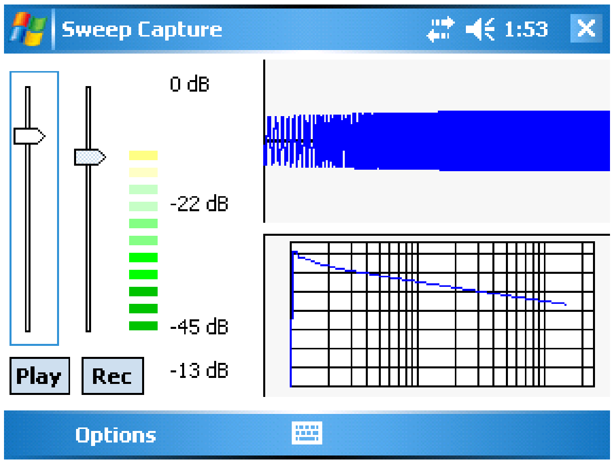


Figure 2: The application on the PDA after a successful recording. In this example a logarithmic sweep was recorded.

A large part of the user interface is dedicated to the plots of the graphical representations for the captured data. The data are analyzed in time (upper display) and frequency (lower display) domain. The user can click on these two boxes and a much larger preview is displayed which provides the ability to store the graphs in various image file formats or zoom in for greater detail.

The Sweep Capture application is written in the managed environment of the .NET Framework whereas the communication with the digital capture card is utilized through native windows libraries [6]. The connection between the managed and native code is achieved through an intermediate library written in the .NET framework with the use of platform invocations (P/Invoke) [7], as shown in Figure 3.

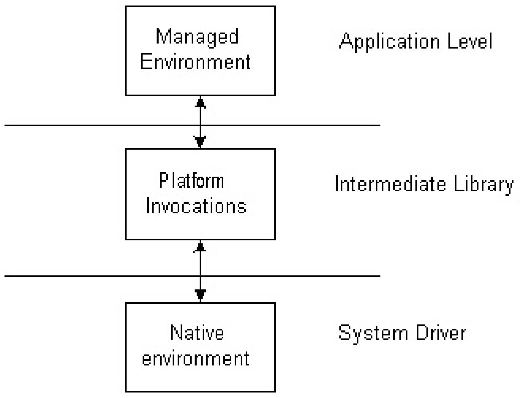


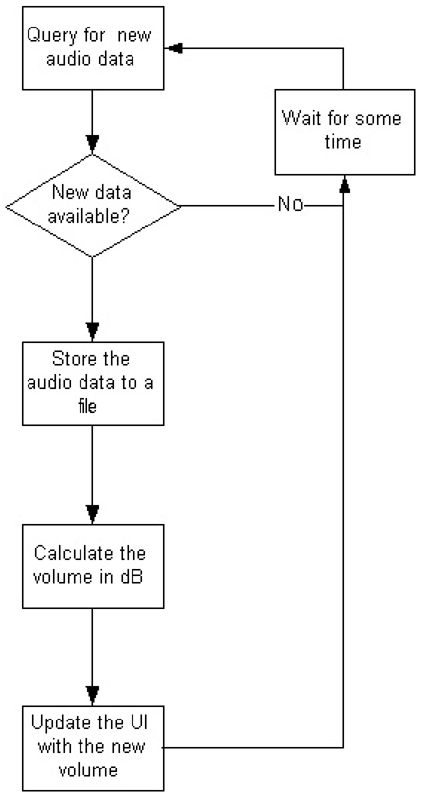
Figure 3: In the PDA, the intermediate library handles the communication between the device driver and the user application.

The implemented intermediate library is responsible for importing functions from the native windows libraries and transforming them into functions that can be used by the .NET framework. This procedure however increases the computational cost and adds an overhead of between 10 and 30 x86 instructions per call [8]. As a result calls to native code are executed only when it is absolutely necessary.

During the startup of the user application on the PDA the intermediate library is initialized and the default values are loaded. The device is inactive and the driver is closed until the user performs a new recording. At this point the application opens the card’s driver and configures the card according to the user selected settings. The driver closes immediately after each successful recording. This open-close procedure is required in order to minimize the dependency of the application on the card. Otherwise, the application would require the constant presence of the card even when not capturing audio data.

The recording procedure is served by a separate thread from the user interface (Figure 4). This thread periodically calls a native system function which is responsible for the communication with the digital capture card. This function retrieves the captured data from the driver's internal memory and stores these data to a wave file. Simultaneously, it calculates the sound level in decibels and updates the user interface with the current and maximum levels. These indications can help the user to remotely correct the reproduction volume on the playback unit.

After the completion of a successful recording, the application automatically initializes the analysis threads. These threads create the graphical representations of the captured data in the time and frequency domain in order to assure the user that the measurement is valid. Since PDAs have limited resources, only the first five seconds of each recording are plotted. This limitation however is not critical for impulse response measurements and furthermore the stored wav file can be later analyzed on a more powerful desktop computer.



Query for new audio data
New data available?
No
Wait for some time
Store the audio data to a file
Calculate the volume in dB
Update the UI with the new volume

Figure 4: The flow chart of the PDA record thread.

3.2. Driver Implementation

A novel driver for the digital audio capture card was developed according to the stream interface driver architecture [9]. All the required functions were implemented as well as a set of system input output calls (IOCTLs) which are used by the user application to setup various card settings.

The digital capture card is equipped with an internal circular buffer of 128 kBytes sufficient for capturing data at 24bit/192 kHz. The driver is responsible to periodically copy the data from the card to the system's temporary memory. This procedure is achieved with the use of an interrupt.

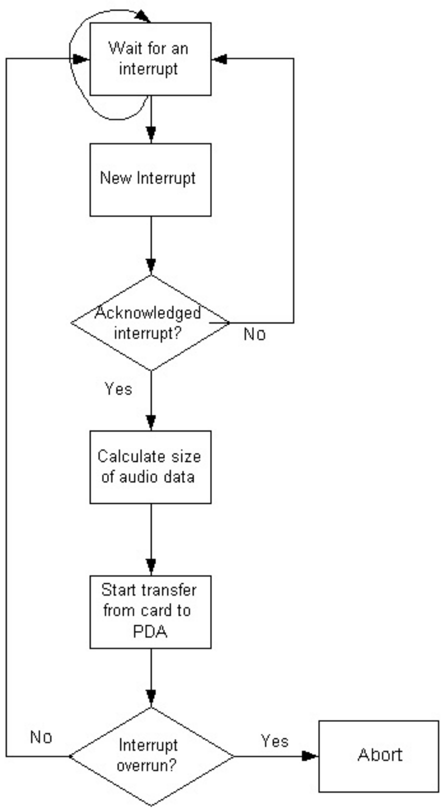


Figure 5: The PDA interrupt service routine. The scope of the routine is to acknowledge interrupts from the card and copy the audio data to the driver's memory.

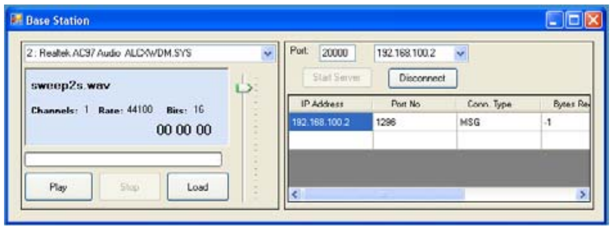
More specifically, the card generates an interrupt each time its internal buffer reaches a predefined size. This interrupt is acknowledged by the system and the corresponding interrupt service routine is called (Figure 5). The routine copies the data from the card's internal buffer to a secondary buffer in the driver's private memory. This procedure must be completed as fast as possible in order to prevent interrupt overruns where an interrupt is acknowledged before the previous interrupt service routine returns. In case of an overrun the recording procedure is canceled and the user is notified about the error.

The data extracted from the card are stored by the driver in a new circular buffer on the virtual memory of the PDA. This memory has a much larger capacity than the internal card's memory which reduces the need of constant transfers. The user application periodically copies the audio data to permanent storage such as a file on a flash card. A problem of synchronization and multiple accesses arises since the same memory space is shared between the driver and the user application. To prevent any corruption the use of an access lock is required. The interrupt service routine locks the PDA's memory before copying new data from the card to the PDA and consequently the Read function waits for the lock to be realized before copying the data from the driver's memory to a file.

Finally, the device is reset and put in low power mode when a successful recording finishes which preserves the battery power. This is accomplished automatically whenever the user application calls the IOCTL that stops the recording procedure.

3.3. Base station application

The base station was built on the .NET Framework and serves as a playback unit and a TCP/IP server. The playback is implemented with the use of the DirectX technology and the wireless communication is processed by windows sockets on the wireless device.



IP Address	Port No	Conn. Type	Bytes Re
192.168.100.2	1296	MSG	-1

Figure 6: The main window of the Base Station application.

On initialization, the application detects any audio and network interfaces available on the host computer. It uses asynchronous sockets to exchange data with the connected client and events to notify the parent thread of new send or receive actions. Both the audio playback and the network communication are executed on separate threads in order for the user interface to be responsive to user input. Furthermore, this procedure makes better use of the resources on the desktop computer especially when combined with a multi core central processing unit (CPU).

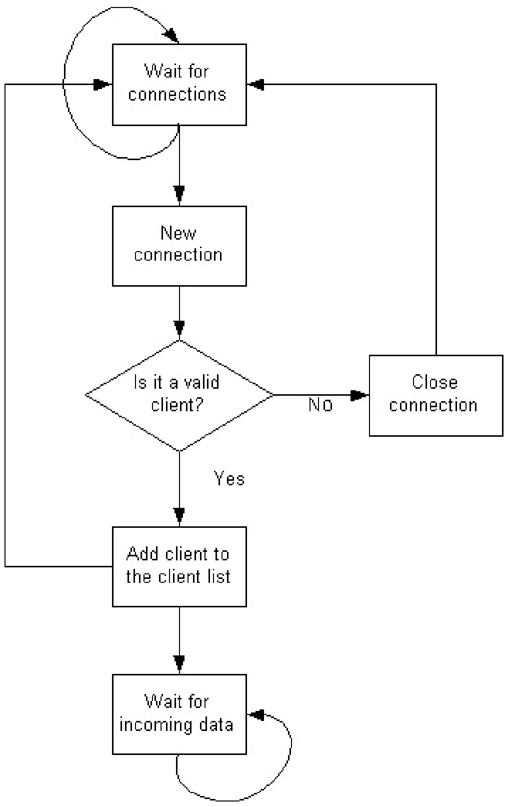


Figure 7: A typical flow chart of a server that waits for clients to connect.

The server initialization includes the setup of the network classes and the beginning of listening for new connections form compatible wireless clients (PDAs). As soon as a new client is connected, the base station is in passive mode and waits for requests by the connected client (Figure 7). Information on the clients is displayed on a list box including each client's IP address and the total number of bytes transferred.

The actual communication between the base station and the PAD is achieved with the use of messages (Figure 8). The messages are predefined sets of words that both the client and the server can understand. The messages are divided in categories that depend on the action that will be performed. The format that is used is the following: <category>#<action>#<value>. The category is the group that the command belongs to. For instance one category could be Playback which includes all actions that control the playback on the base station. The action indicates the desired function in the group that should be executed. An example of an action command is "Volume" which indicates that the volume of the Playback group should be changed. Finally, the Value is the new value that should be set on the playback control. As an example, if the PDA user would like to change the playback volume on the base station a command with the following format will be sent: Playback#Volume#-20 where -20 is the new volume in dB. Not all commands follow this layout. For instance commands like the Play and Stop are transmitted directly as Playback#Play and Playback#Stop.

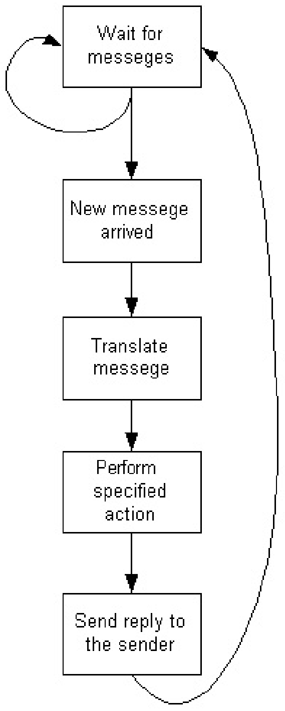


Figure 8: The server waits for messages and when a new message arrives it is translated to a specific action.

Optionally, the server can send a confirmation word after each dictated action has been completed. The confirmation could either indicate success or failure. It is the PDA's responsibility to decide whether to resend the command or abort and notify the user.

4. MEASUREMENTS

During the development of the driver for the digital capture card various tests were conducted to verify the quality of the captured data. A test application was built and with the use of a SPDI/F cable directly connected to a sound card various recordings were compared to the playback data. Moreover, speed tests were conducted in order to minimize the process load of the recording thread. Various scenarios were tested such as the use of fewer interrupts in the card's memory (every ½ and ¼ of the buffer instead of 1/8 that was finally used) and the study of functions that transfer the data from the card's memory to the driver's memory.

The complete system, as described in Figure 1 was tested in a controlled environment. A logarithmic sweep was used as an excitation signal with duration of 2 seconds [10],[11]. The excitation signal was reproduced for 5 seconds for each recording. The recordings were made in a professional laboratory room with dimensions and acoustics close to those recommended by the IEC 268-13 standard for loudspeaker evaluation, i.e. L:7,15m X W:4,60m X H:2,90m. The Reverberation Time was RT=0.368 sec (average).

The captured audio data shown in Figures 10-12, were measured with the microphone approximately 1 meter away from the speaker. The excitation signal was a logarithmic sweep with a duration of 2 seconds (Figure 9).

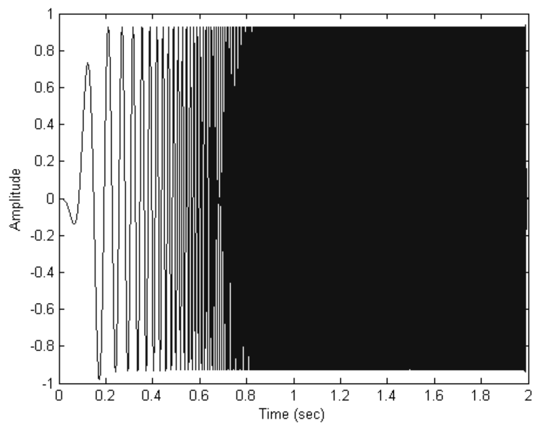


Figure 9: The excitation signal for the acoustic measurement

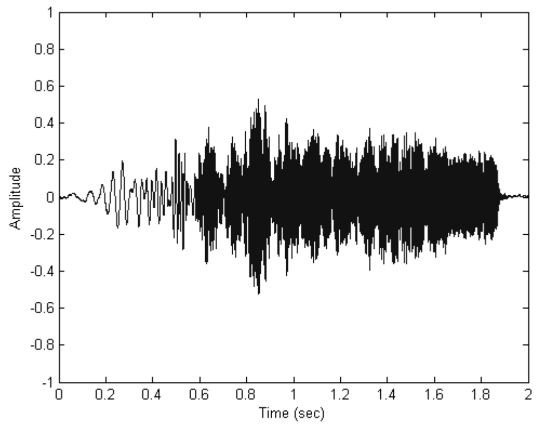


Figure 10: The captured waveform by the PDA.

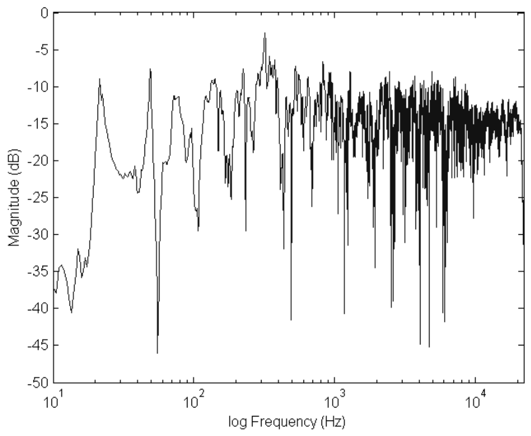


Figure 11: The spectrum of the captured signal.

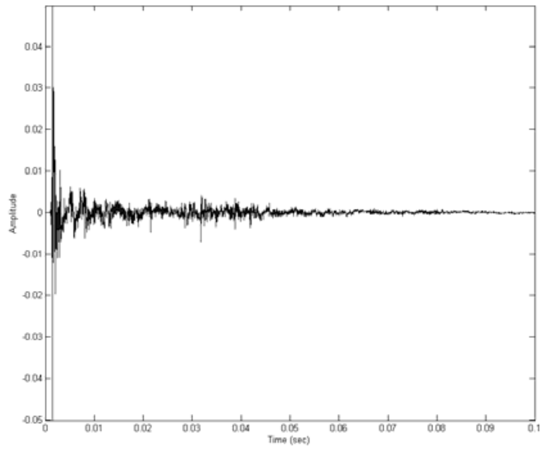


Figure 12: The calculated impulse response

5. CONCLUSIONS

The proposed system utilizes a typical PDA and off-the-shelf hardware to develop a professional-grade audio / acoustic measurement system in order to present the audio engineer with a solution that can be used in an extremely flexible and portable way

The system required the development of various components on different platforms and frameworks. The device driver for the PDA’s digital capture card was developed using native win32 code whereas the applications on the PDA and the base station were built with the use of the .NET framework. Furthermore, knowledge on networking, DirectX, hardware and driver programming was required in order to build the applications and libraries that compose the system. Thus, a wide range of different technologies that interconnect and flawlessly communicate with each other were utilized.

The implemented system proved to be robust and versatile during the test measurements that were performed. Although the setup of the system requires a few more steps than usual, the simplicity of the measuring procedure and the speed with which the engineer can perform measurements in various locations overcomes this excess time needed to deploy the system. Moreover, the accuracy of the recorded data as well as their immediate analysis and preview in each location reduces the risk of incorrect measurements. In general, the audio engineer has the ability to perform more measurements than usual, in more locations while being assured for the quality of each recording.

Finally, the system was implemented in such way that the total cost is kept as low as possible. All hardware used is commercially available and is not specific for the above setup (apart from the digital audio capture card). The audio engineer can choose any wireless hardware as well as any compatible PDA

6. ACKNOWLEDGEMENT

The authors want to acknowledge the contribution of Len. Moskowitz of Core-Sound for the supply of PDAudio-CF digital capture card that was used for the development of the measuring system.

7. REFERENCES

[1] .NET Framework Developer's Guide: “Overview of the .NET Framework” (http://msdn2.microsoft.com/en-us/library/aa735794(VS.71).aspx)

[2] MSDN2: “DirectX 9.0 for Managed Code” (http://msdn2.microsoft.com/en-us/library/ms804944.aspx)

[3] .NET Framework Class Library: “System.Net.Sockets Namespace” (http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/frlrfsystemnetsockets.asp)

[4] Core Sound: “PDAudio-CF” (http://www.core-sound.com/pdaudio-cf/1.php)

[5] Vassilantonopoulos, P. Hatziantoniou, J. Worley, J. Mourjopoulos, J. Merimaa, “The Acoustics of Roman Odeion of Patras: comparing simulations

and acoustic measurements”, presented at the Forum Acusticum, Budapest, August 2005.

[6] CodeGuru: “Managed, Unmanaged, Native: What Kind of Code Is This?” (http://www.codeguru.com/csharp/.net/cpp_managed/article.php/c4871/)

[7] MSDN Magazine: “Calling Win32 DLLs in C# with P/Invoke” (http://msdn.microsoft.com/msdnmag/issues/03/07/NET/)

[8] C# Corner: “Interoperability in .NET” (http://www.c-sharpcorner.com/UploadFile/akchavali/Interoperabilityin11282005052912AM/Interoperabilityin.aspx)

[9] Microsoft Windows CE .NET 4.2: “Stream Interface Drivers” (http://msdn2.microsoft.com/en-us/library/ms894724.aspx)

[10] Farina A., “Simultaneous measurement of impulse response and distortion with a swept-sine technique,” presented at the 108th AES Convention, preprint 5093, Paris, France, 2000 February 19-22.

[11] S. Muller and P. Massarani, “Transfer-function measurement with sweeps,” J. Audio Eng. Soc. , Vol 49, No 6, pp. 443-471, June 2001.