Title

Open Speech Platform: Democratizing Hearing Aid Research.

Authors

Sengupta, Dhiman

Boothroyd, Arthur

Zubatiy, Tamara

et al.

Publication Date

2020-05-01

DOI

10.1145/3421937.3422017

Peer reviewed

Published in final edited form as:

Int Conf Pervasive Comput Technol Healthc. 2020 May ; 2020: 223–233. doi:10.1145/3421937.3422017.

Open Speech Platform: Democratizing Hearing Aid Research

Dhiman Sengupta,
University of California, San Diego, La Jolla, California

Arthur Boothroyd,
San Diego State University, San Diego, California

Tamara Zubatiy,
Georgia Institute of Technology, Atlanta, Georgia

Cagri Yalcin,
University of California, San Diego, La Jolla, California

Dezhi Hong,
University of California, San Diego, La Jolla, California

Sean K. Hamilton,
University of California, San Diego, La Jolla, California

Rajesh Gupta,
University of California, San Diego, La Jolla, California

Harinath Garudadri
University of California, San Diego, La Jolla, California

Abstract

Hearing aids help overcome the challenges associated with hearing loss, and thus greatly benefit and improve the lives of those living with hearing-impairment. Unfortunately, there is a lack of adoption of hearing aids among those that can benefit from hearing aids. Hearing researchers and audiologists are trying to address this problem through their research. However, the current proprietary hearing aid market makes it difficult for academic researchers to translate their findings into commercial use. In order to abridge this gap and accelerate research in hearing health care, we present the design and implementation of the Open Speech Platform (OSP), which consists of a co-design of open-source hardware and software. The hardware meets the industry standards and enables researchers to conduct experiments in the field. The software is designed with a systematic and modular approach to standardize algorithm implementation and simplify user interface development. We evaluate the performance of OSP regarding both its hardware and software, as well as demonstrate its usefulness via a self-fitting study involving human participants.

dhimnsen@eng.ucsd.edu .

Keywords

hearing aids; open source; digital signal processing; real-time; human study

1 INTRODUCTION

Hearing plays a vital role in how we interact and socialize with the world. When a person's hearing is impaired, it has a profound impact on how the person interacts with their surroundings and often leads to isolation from the world [18], resulting in substantial economic and societal costs [27]. In 2015, 28.8 million people in the United States suffered from hearing loss, but only 1 in 4 affected use hearing aids [25]. Even though hearing aids have shown to benefit the hearing aid users significantly [40], the lack of adoption is due to self-reported hearing performance, technology commitment, and the socioeconomic and health status of the hearing-impaired persons [37]. Researchers are developing novel ways to improve hearing aids in order to increase the adoption of hearing aids. Unfortunately, the lack of open-source, standards-based hardware platform prevents cutting-edge academic research from prospering and making its way into commercial use [23].

In 2014 the National Institute on Deafness and Other Communication Disorders (NIDCD) organized a workshop to incubate research that goes "beyond what is widely done today" and to identify barriers to commercializing academic research [23]. The workshop found that audiologists and researchers lack the tools required to quickly and cheaply test new ideas outside the lab, which makes it hard to know how their research will translate to everyday use.

The status quo of hearing aid research faces a few key barriers. First, despite abundant innovative research on improving hearing aids [3, 10, 12, 15, 17], the produced results and artifacts are hard to compare and reproduce [28], mainly due to lack of a standardized research hardware platform. Second, the disparate hearing aid platforms from major manufacturers contain proprietary, closed-source software. These barriers make it difficult for audiologists to understand the details of algorithms on-board when interpreting the results produced by these devices. Furthermore, the interfaces that accompany hearing aids today enable researchers and end users to control features such as volume, but lack the flexibility that researchers need to test novel techniques [22]. The lack of decent amenable interfaces restricts the use of these devices, affects users' experiences, and prolongs the cycles of experiments and studies for audiologists [39].

This issue calls for an open-sourced research system equipped with reliable hardware that meets the industry-standard as well as software that is modular and complete, in order to spur and facilitate hearing aid research.

To this end, we designed and developed the Open Speech Platform (OSP), a hardware and software solution to bridge the gap. OSP is an open-source, portable, and extensible platform for hearing health care research. The software bears a modular, systematic architecture with standard functionalities required for hearing aid research. At the same time, we built the portable hardware with clinical-grade behind-the-ear receiver-in-canal (BTE-RIC) hearing

aids using commercial-of-the-shelf components with custom printed circuit boards and plastics. We also designed the OSP with feature-rich web graphical user interfaces that allow for the capability to control and modify the internal components and their parameters in real-time, which could potentially promote wider adoption of such a system and facilitate research and studies.

We summarize our major contributions as follows:

- The design and development of a multidisciplinary platform, the Open Speech Platform, with open-source software and hardware implementation for the advancement of hearing health technologies;
- The inclusion of interfaces (both graphical and programmatic) to users and researchers for a better experience and use in studies and experiments;
- A systematic, modular software design to facilitate and standardize algorithmic workflow composition;
- Evaluation of both the hardware and software performance, together with a usability study involving human participants.

2 RELATED WORK

In this section, we review related papers and products on two lines of work. First, we survey current standardized hardware and software frameworks for hearing health care. Then we examine works on controllability and amenability in hearing health care.

2.1 Standardized Hardware and Software Frameworks

There has been extensive research on developing hearing aid algorithms [3, 10–12, 15, 17]. However, studies have shown that various hearing aid models, even from the same manufacturer, behave differently with regards to how they modify the signal in different conditions even if set up with the same configuration (e.g., signal-to-noise-ratios and presentation levels) [28]. Furthermore, these disparate hardware platforms often contain proprietary software, over which researchers have little to no control. Altogether, these make it hard to scientifically compare, or even to reason about the results from different research obtained using these hardware platforms. Seeing the need for standardized hardware, Tympan developed a low-cost open-source portable hardware platform for hearing aid research and development with very intuitive software development tools [38]. However, it is based on a small embedded microprocessor which lacks the processing power for more advanced and demanding algorithms such as least mean square based adaptive feedback cancellation while processing six wide dynamic range compression bands.

Similarly, UT Dallas developed a smartphone-based open-source research platform for hearing improvement studies as a smartphone-app solution for Android or iOS running on various hardware [19]. However, as shown in their study [19], only the iPhone models demonstrated a latency of less than 10 ms, which meets the real-time processing often required by hearing aids algorithms. However, the latency will only worsen with more demanding algorithms.

HorTech and the University of Oldenburg released openMHA [8, 13] in 2017, which is a software only solution based on the closed-source commercial HorTech Master Hearing Aid software[1]. Open-MHA is a development and evaluation platform with a vast DSP library that is capable of executing hearing aid signal processing in real-time on standard computing hardware with low latency (<10 ms) between the sound input and output. However, openMHA requires an additional crafted control interface if users need a GUI control of the system. It is also not clear if openMHA can take full advantage of multiple-core systems for the signal processing because, as stated in [8], the processing is done in one real-time thread and another thread is used for control. Our platform, on the other hand, was designed with the ability to use multiple cores for efficient use of resources in both embedded systems and standard computing hardware.

To overcome these issues, we develop the Open Speech Platform (OSP), which is an open-source hardware-software co-design to provide a standardized platform for developing and evaluating hearing aid research. OSP also provides standardized metrics for evaluating algorithms (e.g., w.r.t gain, distortion, latency, etc.) across studies [28, 31]. Finally, with the standardized hardware and software interfaces, OSP makes it easier to develop algorithms and conduct experiments, to perceive and compare results, and to adopt research artifacts between studies. This allows researchers to integrate the findings of other studies in the community into their research.

2.2 Controllability and Amenability

It has been shown that giving control to the users, even by allowing them to be able to monitor or control their hearing settings, makes users feel more in control of their hearing loss [22]. Unfortunately, while many studies in audiology today utilize complex signal processing algorithms [16, 26, 29, 31], many do not incorporate a user interface. Boothroyd Mackersie et al. in their Goldilocks study used a custom DADiSP[2] app in a self-fitting experiment to see if users can find their way towards their personal prescription from a generic starting point [2, 20]. The DADiSP interface allowed Boothroyd Mackersie et al. to rapidly prototype, however, DADiSP is restricted to running only on standard desktop environment.

For this reason OSP provides a web-based interface to allow user control, which makes implementing experimental interfaces a matter of editing HTML, CSS, and JavaScript from a base template, rather than requiring platform-specific knowledge (such as iOS or Android apps).

3 OPEN SPEECH PLATFORM

3.1 Design Philosophy

Audiologists and researchers will benefit from a research system with several essential properties [24]. First, the system should be entirely open-source, both hardware and software, which allows for a standardized experimental workflow enabling researchers to

[1] https://www.hoertech.de/
[2] https://www.dadisp.com/

evaluate and compare research results as well as artifacts. Second, the system should contain a user interface that allows researchers to bring patients into the loop, is simple to edit, and exposes the internal components and details (e.g., signal processing techniques) of the hearing aid for transparent control. Third, the system should contain a standard tool for conducting experiments “in the wild.” Finally, the system should be able to both accurately and reliably reflect the acoustic properties of hearing aids commonly used today. The rest of this section will discuss, based on this design philosophy, the design of the software, the design of the hardware, and the evolution of the Open Speech Platform (OSP). All software and hardware described in this paper is open-source and available at https://github.com/nihospr01/OpenSpeechPlatform-UCSD.

3.2 Software Design

At the core of OSP software, Figure 1, is the real-time master hearing aid (RT-MHA), which is in charge modifying the audio stream in real-time. At the same time RT-MHA needs to change the behavior of the audio processing when it receives commands from the external control interfaces, such as the embedded web server (EWS). The EWS hosts a suite of web apps which allow the user to control RT-MHA through graphical user interfaces (GUIs). We further elaborate on the design of each of the components.

3.2.1 RT-MHA.—When designing the RT-MHA framework, our goal is to create a modular environment that abstracts the details of the real-time hearing aid algorithm (i.e., common functionalities such as wide dynamic range compression (WDRC), speech enhancement, feedback cancellation, etc.) into simple, extensible, and user-friendly application programming interfaces (APIs) . As illustrated in Figure 2, RT-MHA hosts a few key components: 1) a central Real-Time Engine that handles all the callbacks issued by PortAudio and orchestrates the processing of the audio; 2) an auxiliary Parser interface to parse the input from users through the programmable or graphical interfaces and set the parameters of modules and algorithms in the hearing aid; 3) a external connection interface for communicating with GUIs; 4) a command line interface (CLI) tool for debugging purposes; 5) a PortAudio module that issues callbacks for further processing when audio data arrive.

Real-Time Engine.: The core of the RT-MHA framework is the Real-Time (RT) engine, a C++ development environment that provides the developers with a real-time modular environment to develop their hearing aid algorithms. We envision that implementing hearing aid algorithms in the RT engine would be as simple as to compose a structured ensemble of library modules. An example of this is the reference hearing aid algorithm that we provide as a part of the OSP system, as illustrated in Figure 3. This reference design is based on Kates work [11] and incorporates the fundamental algorithms that are necessary for hearing aid. The following briefly describes the function of each of the components in the reference design, for more details see [7] :

- *Resample 3:2 and 2:3*: These blocks re-sample the input stream from 48 kHz to an output at 32 kHz and vice-versa.

- *sub-band-N* : This block is a band-pass filter that separates the incoming signal into one of N different sub-bands for further processing (6 bands in the reference design).
- *Speech Enhancement*: This block uses statistical data analysis to detect the presence of background noise and suppresses it from the output signal.
- *WDRC-N* : The Wide Dynamic Range Compression (WDRC) block takes the input signal and a sub-band and remaps the signal from full scale to a restricted range.
- *Feedback Cancellation*: This block uses the estimated transmission path between the BTE-RIC's output speakers and microphones to reduce ringing from feedback.
- *Feedback Path Estimation*: This block periodically estimates the transmission path and updates the filters coefficients in the Feedback Cancellation block.

The key to creating a modular real-time environment starts with the basic building blocks, which, in our case, are the *library modules*. Therefore, we developed a template on how to design a library module (see Listing 1). The template dictates that the library module must have a function for setting the parameters, a function for getting the parameters, and a function for processing the real-time audio data. This template requires each library to have a private data structure for communicating data between the functions using a global shared pointer. The global shared pointer points to the currently valid data structure, and only the set parameter function can update the global shared pointer. When the set parameter function, for example, needs to update the parameters, it creates a new data structure. It then atomically replaces the global pointer to point to the new data structure. As for the other two functions, they can only atomically load the data structures and read the values contained in them. This method of one-writer-multiple-readers allows us to create a lock-free environment with minimal interaction between the functions, which ensures that non-real-time interactions will not impact real-time performance. This style of coding is pervasively used in the audio industry [4], and we deem it the best model when developing the OSP system. The benefit of adopting such a design is that modules developed using this template are modular and capable of uninterrupted real-time processing.

On top of providing a modular environment, the RT engine can create three domains for audio processing to spread the processing over three real-time threads. The first domain is the binaural domain, which processes the algorithms that require a signal from both ears. The second is the left domain, which deals with algorithms that modify signals for the left ear. Similarly, the right domain is for the right ear. The three domains allow for more efficient use of a multiple-core system, like our portable device in subsubsection 3.3.3.

Parser.: The parser is the gatekeeper when deciding what parameters are accessible to users through the programming or graphical interface in OSP. Exposing the internal parameters to the users would provide them with transparent control over the internal components of an algorithm. The algorithm developer will decide what parameters to expose in the user-facing interface by modifying the parser for their RT engine. Based on the list of

parameters to expose, the parser would then create a JSON string-based API that contains these parameters, which is how the services such as the EWS (which we shall explain shortly) would interact with RT-MHA.

External Control Interface (ECI).: The ECI module is the process in the RT-MHA framework that listens on a TCP/IP port for incoming commands from a control application. The parser serializes the parameters into a human-readable JSON format, which the ECI module then sends to the control application when requested. When the control app needs to change the parameters, it will send a human-readable JSON string of the parameters with the new values back to the RT Engine through the parser via the ECI module. The parser would update the parameters atomically without interrupting the RT Engine. This mechanism allows for any type of GUI to interface with RTMHA.

Command Line Interface (CLI).: For debugging and rapid prototyping of RT-MHA, we designed a simple command-line interface that can interact with RT-MHA through the parser. The CLI provides the developer with a quick and convenient way to test both the Parser and RT-MHA without the need to connect through a separate control application.

PortAudio.: In order to create a device-(and operating system) agnostic framework for real-time master hearing aid development, we need to abstract the low-level details about how the algorithms interact with the hardware. Through experimentation, we discovered that PortAudio[3], open-source audio I/O library, meets all of our design needs, like providing a comprehensive abstraction of the hardware, being open-source, supporting Linux/macOS/Windows, and allowing for low latency input/output.

3.2.2 EWS Framework.—

A common choice for providing a means for the user to interact with a system and control the components is to develop phone applications (e.g., on Android or iOS). However, for a development platform such as OSP that is intended for a broad spectrum of users from researchers to audiologists, a phone application will dictate a limited set of devices people could use to interact with RT-MHA. According to Xanthopoulos et al. [41], web apps are becoming an increasingly better option for mobile app development because there are many libraries, such as HTML5 and JQUERY, which simulate native app functionality. Plus, the development time is much shorter and less complicated than for native apps. Therefore, we choose to build an environment that would allow us to create web apps — browser-based applications downloadable through the web. This way, these web apps can be used from any browser-enabled device.

To this end, we have incorporated an Embedded Web Server (EWS) into our OSP platform, which hosts a suite of web apps, like the researcher page in Figure 4. These web apps are responsive to the form factor of the device. The responsive web apps are platform-agnostic [30] allowing users to control the RT-MHA from any web-enabled device.

EWS is based on modified Linux-Apache-MySQL-PHP (LAMP) architecture. The Apache server has been replaced with a light-weight HTTP server provided by the Laravel

[3] http://www.portaudio.com

framework. MySQL has been replaced by SQLite for simplicity and to minimize resource usage.

The web-apps are implemented primarily in HTML5, CSS, and JavaScript. This way, incorporating web-apps reduces the barrier to creating tools to control the RT-MHA for hearing aid application developers.

3.3 Hardware Design

This section describes the design of three hardware devices we developed: the hearing aid, the lab system, and the portable system.

3.3.1 Hearing Aid Design.—We designed and developed a clinical-grade hearing aid, based on the most common hearing aid form factor called behind-the-ear receiver-in-canal (BTE-RIC). The design of this hearing aid is simple yet effective; it has one MEMS microphone for the input and one clinical-grade receiver for the output, as shown in Figure 5. Internal to the hearing aid, there is a pre-amplifier circuit, which amplifies the microphone signal before sending it along the wire in order to make the signal noise tolerant. The receiver is connected directly to the wire and requires no amplification in the hearing aid.

The challenge in developing the hearing aids, however, is designing the plastics. The shape and size of the plastic have a significant impact on the acoustic quality of the hearing aid. The cavities created by the plastic shell resonate at particular frequencies, which causes significant feedback between the microphone and the speaker to overwhelm the Adaptive Feedback Cancellation algorithm. After understanding how the plastic shell affects the acoustics, we designed the BTE-RIC hearing aid shown in Figure 5, which performs on par with other commercial hearing aids, Table 1.

3.3.2 Lab System.—During the development of the hearing aids, we created a lab-based system, which allows the researcher to use any computer installed with either Linux or macOS. Our lab system, Figure 6a, is comprised of a laptop, an off-the-shelf audio converter, a break out board (BoB), and a hearing aid we designed. The audio converter box is a Focusrite box with a Thunderbolt connection to the Mac. Finally, the BoB is a custom printed circuit board (PCB) for amplifying both the signal coming from the microphone to the Focusrite box and the signal transmitted from the Focusrite box to the receiver. The BoB ensures that signal feed into the Focusrite, and the signal that goes to the receiver is at the correct level.

Even though in our setup, we have the BoB connecting to the Focusrite box, the BoB can connect to any audio conversion box. Therefore, this system allows researchers to design and develop on any computer while having access to hardware, which is equivalent to commercial hearing aids.

3.3.3 Portable System.—In order to enable experiments in the field, we also designed a portable version of the system , Figure 6b, which is composed of a DragonBoard 410c, a single board computer (SBC), a custom printed circuit board (PCB) daughterboard, and

a 23 Wh rechargeable battery. All of these are packaged into a 3D printed shell, which we envision users would hang around their neck while using the hearing aids.

The first step in designing the portable system was choosing the embedded computer, which would be the foundation of the portable system. We decided on the DragonBoard 410c SBC because it achieves a balanced trade-off between computational power and power efficiency, with an active hobbyist community to support its use and development. The DragonBoard 410c consists of a quadcore ARM A53 chipset with 1 GB of RAM and 8 GB of memory. The board contains WiFi, Bluetooth, and GPS. The DragonBoard 410c also has a built-in audio codec.

In order for the hearing aid to interface with the codec, we needed to design a daughterboard with similar functionalities as BoB from the lab-based system. Therefore, we designed a PCB, Figure 6c, that interfaces with the DragonBoard 410c SBC, which amplifies the signal from the microphone and to the receiver. The board also contains a mute switch for safety measures and a debug port allowing researchers to access the portable device over USB. Lastly, the daughterboard can recharge and operate from a 23 Wh battery, which gives the system around 12-hour battery life.

Lastly, the portable system runs Debian Linux, which makes porting code between the lab system and the portable system relatively easy. This means that we can develop on the lab-based system and then test on the portable system when the code is ready.

3.4 Evolution of the OSP

The design of OSP went through three major cycles, as we interacted with and received feedback from collaborators.

3.4.1 First Cycle.—Starting from the need for a portable system to bridge the gap between academia and industry, which originated from the NIDCD workshop [23], our first step to designing OSP was to gain a better understanding of the gap. We embedded ourselves in a few research labs focused on hearing health care to understand the problem better; mainly, the Auditory Research Lab (ARL) at San Diego State University (SDSU) headed by Professor Carol Mackersie.

ARL is interested in conducting long-term experiments using a mobile app for their “Goldilocks” self-fitting experiment [2, 20]. They had implemented and validated a proof of concept for “Goldilocks” using a calibrated headphone connected to a computer, which would play pre-processed sounds as the user interacted with a graphical user interface built using DADiSP. We noticed two critical limiting factors that could be improved further. The first is the lack of hardware that can interface with a computer and mimic the form factor used by commercial hearing aids. The second is the use of pre-processed data rather than live audio since it is easier and more reproducible to use pre-processed data. However, pre-processing the data limits the exploration space due to the amount of pre-processed data that would need to be saved as the dimension of the parameters they explore grows. These two issues defined our first milestone, i.e., to develop hardware that mimics a commercial hearing aid on a lab-based system and software that processes the data in real-time.

At the end of our first development cycle, we designed and developed the BTE-RIC hearing aids and the lab-based system described in Section 3.3.2. Also, we developed the first version of the RT-MHA software, which included an Android app as the user interface. Many smaller iterations occurred during this development cycle, where collaborators at ARL tested the system and provided us both empirical and subjective feedback based on their experience, which were of great help. At the end of the first milestone, Mackersie et al. were able to conduct a user study on self-fitting – Goldilocks – using the OSP lab system [21].

3.4.2 Second Cycle.—Since the Goldilocks study heavily relied on the self-fitting app, our focus next was on the user interface for OSP. Therefore, the second major cycle was focused on how to improve the Android app built earlier. During this process, we realized that, as a development platform, using an Android app greatly restricts its use and development to a particular framework, i.e., Android. Our goal for the second milestone was to overhaul the user interface of OSP while enabling our collaborators to conduct studies in the field.

The second development cycle culminated in the introduction of the EWS in the OSP software stack (Section 3.2.2) and the development of the portable hardware system (Section 3.3.3). Internally the development time required to iterate on the user interface decreased significantly once we migrated to the EWS version of OSP, mainly attributed to the smaller learning curve. The biggest hurdle during the second development cycle, however, was designing the daughterboard that interfaces the BTE-RIC to the single-board computer (SBC). We had to design the daughterboard to compensate for flaws in the SBC's audio circuits. We iterated on daughterboard design a few small cycles before reaching an acceptable version.

3.4.3 Third Cycle.—A few of our other collaborators are interested in integrating their algorithms with our platform, e.g. using Open-MHA software on OSP. However, we realized that the first version of the RT-MHA software was not modular and made it difficult to compare different artifacts. Another key issue raised was that, when interacting with RT-MHA using the GUI, users would notice loud clicks and pops, which indicates that the real-time property were violated. Therefore, the third milestone was to refactor the RT-MHA software.

Our third development cycle introduced the RT-MHA framework in Section 3.2.1, which allows for a modular real-time development environment. The rest of this paper evaluates the OSP system delivered at the end of the third development cycle, which includes both an empirical evaluation and a subjective evaluation with hearing aid users.

4 EVALUATION

In this section, we empirically evaluate the usability of the OSP platform concerning three different criteria. First, we evaluate the hardware in terms of how well it corrects for the end users' hearing loss. Then we determine whether the latency caused by the audio processing by the platform meets the requirements set forward by the series of studies called

the Tolerable Hearing-Aid Delays [32–36]. Finally, we evaluate the impacts of the user interacting with the platform on the real-time performance of RT-MHA.

4.1 Hearing Hardware Evaluation

We first examine the quality of the hearing aid hardware developed for both the lab-based system and the portable system by comparing them against four commercially available hearing aids.

4.1.1 Setup.—To test the quality of the hearing aids, we use a test created by the American National Standards called ANSI 3.22–2003 [9]. The test mainly is used to verify the claims made by the manufacturer of the hearing aid. Therefore, we compared our lab and portable setup against four commercially available hearing aids.

In order to perform this test, we used the Verifit 2 test equipment by AudioScan[4]. It isolates the hearing aids in a quiet environment and uses its calibrated speakers and microphones to measure the ten different metrics of the ANSI 3.22–2003 test. During our testing, we tested two different receivers in our systems. The first receiver, receiver X, is a high bandwidth Knowles receiver for mild to moderate hearing loss and has a higher frequency fidelity. On the other hand, receiver Y is a high power Knowles receiver that is used for users with moderate to profound hearing loss.

4.1.2 Results.—The first four rows of Table 1 shows that both the systems are able to correct for the same amount of hearing loss as any of the commercial hearing aids we tested. According to the rest of Table 1 both the lab system and portable system are within specification with the only exception being the equivalent input noise for the OSP Portable System. We identified the codec on the Dragonboard 410c board as the offending hardware that causes the high input noise. The audio codec is part of the power management IC (PMIC) on the Dragonboard 410c board, and therefore the switching noise of the PMIC is picked by the microphone circuitry, which is the reason for the higher than usual noise. However, as we will see in the usability study in section 5, most hearing-impaired users were either not able to notice the noise or not too bothered by it.

This test verifies that the hearing aid we built, along with the underlying software, is close to the specifications of commercial hearing aid, promising its adoption and use in research experiments and studies.

4.2 Processing Latency of OSP

In a series of studies called Tolerable Hearing-Aid Delays [32–36], audiologist and psychologist determined that 20–30 ms is the maximum latency a user of a hearing aid can tolerate. Latency refers to the time it takes audio to be captured by the hearing aid and played back into the listeners' ears. The longer the latency, the more uncomfortable it becomes to use the hearing aid; according to these studies, at the 20–30 ms mark is when

[4] https://www.audioscan.com/en/verifit2/

users start to notice the latency becoming unacceptable. Therefore, we need to design and ensure that the combination of the hardware and software is within the acceptable range.

4.2.1 Setup.—To evaluate the latency of OSP, we consider both the hardware and software. The devices were set up in loop-back mode, where the output was directly fed into the input. Then we played a sound through the device and recorded the input of the device at the same time. Comparing the input and output files, we can determine the latency of the hardware by calculating the latency between the two files. Next, we performed the same test except for playing the sound through the RT-MHA algorithm and then recording the output, helping us determine the software attributed latency. By adding the combination of the recorded latency, we can determine the overall end-to-end latency of the device when in use.

4.2.2 Results.—The results in Table 2 indicate that the reference design of RT-MHA can be run on either the lab system or the portable system and is still below the 20–30 ms latency threshold. This verifies that the current hardware and software design can be used in clinical trials. The results also show that there is plenty of resource in terms of latency for more sophisticated algorithms to be implemented in RT-MHA.

4.3 Impact of User Interaction on Real-Time Performance

The usability of a hearing aid system is rooted in its real-time performance. RT-MHA receives one data packet of audio every 1 ms and needs to process it before the next one arrives. If RT-MHA misses its deadline to process the audio, the user would hear “clicks” and “pop” sound. Audio artifacts are acceptable once in a while; however, continually hearing these artifacts can make hearing aids unusable. Algorithm developers have to ensure that their software can meet real-time requirements. Developing real-time software usually involves anticipating how the interaction between the non-real-time processes will impact the real-time performance. We shall demonstrate next that the RT-MHA framework in OSP has successfully decoupled the effects of non-real-time processes on the real-time algorithm. This means that neither the UX designer nor the algorithm developer has to worry about how the software they develop will impact the other.

4.3.1 Setup.—In this experiment, we set up the portable device with two BTE-RIC hearing aids and enabled all of the hearing aid functionalities described in the reference design. The software was instrumented with real-time profiling code, which calculates the *time* taken for RT-MHA to complete its processing. The real-time profiler allows us to determine whether the master hearing aid (MHA) algorithm can meet the soft real-time requirements using the resources available on the system. It also enables us to evaluate the effects of a user interfacing with the device on the real-time resources that RT-MHA requires. Notably, we explore the impacts of user interaction with the platform, through the EWS, on the real-time master hearing aid algorithm.

We started by running all of the algorithms enabled in the reference design and profiled the real-time resources used by RT-MHA for an hour as the baseline. Then we connected a secondary device via Wifi broadcasting from the device. The secondary device has a python

script running, which randomly changes the parameters of the reference design running on the device at a particular interval to mimic a user interfacing with the device. In our evaluation, we chose three different intervals — 30 seconds, 10 seconds, and 1 second, ran the experiment for an hour for each interval, and measure the execution as mentioned earlier to evaluate the performance.

4.3.2 Results.—Figure 7 summarizes the evaluation. In Figure 7, we plot the probability distribution (also termed as a probability density function, PDF) of the time it takes the reference design to complete processing one clip of data. RT-MHA receives one clip of data every 1,000 us, which means that the time that the reference design has to complete the processing is bound to a hard upper limit of 1,000 us. Therefore, to assess whether an algorithm is real-time, we need to inspect the worst-case scenario. As we can see from Figure 7, the worst-case scenario happens at around 750 us for all four of the runs.

First, this indicates that the reference design can finish computation ahead enough of the deadline. There is also plenty of headroom, which indicates that we can expand the reference design to include more algorithms while still maintaining real-time performance. The second and most relevant information conveyed here is that users' interaction with the system through the EWS has no impact on the real-time processing; therefore, this validates the software design decisions for the RT-Engine in Section 3.2.1.

5 USABILITY STUDY

In the previous section, we empirically showed that the OSP platform well meets the industry requirements on acoustic features and processing capability. OSP promises a usable hardware basis for hearing health care research. In this section, we shall present a subjective evaluation of the OSP system to determine the acceptability of the OSP device to a person with hearing loss through a structured interview as part of a pilot field study evalutating OSP using the "Goldilocks" app. This study was designed and ran by our collaborators at the Auditory Research Lab (ARL) at San Diego State University (SDSU) [1].

The objective of this evaluation was to figure out the usability of the OSP from a hearing aid users point of view in terms of 1) aesthetics, 2) sound quality, 3) ease of use, and 4) whether they are willing to use the device for extended periods of time outside a lab for research purposes. The users were interacting with the OSP system throughout the "Goldilocks" pilot study, which lasts for two and a half hours. A fraction of the study had the users use the OSP device outside the lab environment, giving the users a diverse experience with the OSP system. At the end of the study, we collected the participants' feedback and thoughts on the OSP using a structured interview.

In the rest of this section, we will describe the participants and the equipment used in the study. We then outline the procedure that the participants followed. Finally, we present and discuss the results and findings of the study.

5.1 Participants

The Institutional Review Board at SDSU approved the study before data collection and written informed consent was obtained from all participants. Twenty one people (twelve male, nine female) were recruited for the study with an average age of 75.1 years (std = 13.9 years). All participants have their own hearing aids, and the group average for how long they have owned a hearing aid is 7.4 years (std = 7.0 years). The group average for the amount of time in any given day the participant uses their hearing aid is 10.9 hrs/day (std = 5.3 hrs/day). All participants were compensated for their time.

5.2 Equipment

The experiment was conducted in a room equipped with a sound level meter, recorder, a wifi hotspot, a Verifit device, an OSP portable device with hearing aid, a computer, an amplified speaker, an otoscope, the user's smartphone and some extra batteries. The sound level meter is used to calibrate the amplified speaker. We use the recorder to measure the environmental noise when testing outside of the lab. The portable OSP device is the hearing aid device that the participant will be using. The WiFi hotspot connects the computer, the user smartphone and the portable OSP device together. The Verifit device is used to program the initial starting conditions on the portable OSP device and measure the fitting of the hearing aid. There is a computer for the researcher to program different starting conditions on the portable OSP device and to play stimulus through the speaker. The amplified speakers play the audio stimuli to the participant who will be sitting in front of it during the lab section of the trial. The otoscope is there to check the participant's ears before inserting any object in the canal. The user phone is how the user will interact with the self-fitting app running on the portable OSP system, via the wifi hotspot (Figure 8). Finally, there will be extra batteries just in case if one of the battery operated device runs out of power.

5.3 Procedure

The OSP usability using "Goldilocks" pilot study occurred in individual sessions over the course of 7 months. Participants were numbered in the order they participated in the study, and experimental conditions were assigned based on odd or even number. First, all participants completed a consent form and intake form. Next, the audiologist measured the participant's pure tone audiometry (PTA)[5], a metric that quantifies the patient's hearing loss. The research then programmed two sets of starting condition parameters on the OSP platform for the SFP study. The first was called "NAL-NL2", which uses the NAL-NL2 prescription [14] software to convert PTA into the different parameters used by the hearing aids. The second is called "GENERIC", which is the same setting for all participants.

The participant was then handed the phone with the self-fitting app running and was asked to choose either "NAL-NL2", if their participant number was odd, or "GENERIC", if their participant number was even, for the starting condition. Then the research would play a speech stimuli through a speaker mounted in front of the participant and the participant was asked to adjust their aid using the app. At the end of this period, the new hearing aid fit was

[5] https://www.asha.org/public/hearing/Pure-Tone-Testing/

saved as “NALQ1” for odd participants and “GENQ1” for even participants. The trial was repeated again this time starting with the other starting condition.

After finishing the second trial, the participant was asked to choose “NAL-NL2” as the starting condition. Then the researcher would play a speech plus noise stimuli through the speaker and the participant was asked to adjust their hearing aid. The outcome from this trial was saved as “NALN1”.

The researcher would ask the participant to select the starting condition “NALQ1” and would attach an external recorder on the subject. Then they took the participant out of the lab and into a public setting. There the research maintained a conversation with the participant as they adjusted the hearing aid. Once the participant completed this task they saved this fitting as “OUT1”.

The participant was brought back into the lab and the first three experiments were run again, except this time the results were saved as “NAL-Q2”, “GENQ2” and “NALN2”. This point marked the end of the self fitting portion of the study.

Next, the researcher would evaluate the outcome of the self-fitting by having the participant take a speech recognition test four times once using the “NALQ2” setting, once using “NALN2”, once with their own hearing aids and once without any hearing aids. After this, the researcher would measure the amount of correction the hearing aid provides under all of the conditions saved during the trial. This is done by taking the sound quality measurement in the participant’s ear canal using a probe tube while they wear the OSP hearing aids. The researcher records the in ear measurements for all seven saved fitting conditions, as well as, with both the participants own hearing aid and without any hearing aids. This marks the end of the measurement portion of the study.

Lastly, each participant was debriefed using a semi-structured interview. The participants answered questions about the usability of the device in terms of: aesthetics/form, sound quality, long-term usability and ease of understanding the app instruction flow. We next analyze the users’ responses and present our findings.

5.4 Interview Results and Findings

About 56% of participants thought that the aesthetics/form of the portable device were either “good” or “very good”. Other participants commented about the size of the device being too big or the device awkwardly swinging when they walk. We learned that 80% of participants thought the acoustic quality of the portable OSP was “good” or “very good”, while 68% of users thought that the sound quality was at least as “good” or even “better” than their normal hearing aids. While 90% of users said that the ease of use of the “Goldilocks” interface implemented in the EWS was “good” or “very good”. They found the application to be responsive and simple to use both indoors and outdoors. About half of the participants said that they would wear the portable OSP system in a multi-week research experiment. The participants that are willing to do the long-term experiment responded that on average they would be willing to use the device for 3 hours a day for 4 days a week for 2–3 weeks.

We found these results to be encouraging and somewhat expected. Despite receiving commentary from many participants about the portable system being too bulky, over half of the participants said they would possibly or definitely be interested in participating in further research using the portable OSP. We were pleased to learn that most users thought the sound quality and overall experience was good enough that they would be willing to use the system for a research study. This is encouraging because such a study would allow our collaborators to answer very interesting questions in their research. About 90% of participants commented that a big success of this pilot study was the user interface, and a couple of participants noted that the process of self-adjustment was just as easy outdoors as it was indoors. The opportunity to run experiments using OSP in the wild with hearing-impaired users will give researchers a more realistic understanding of their patients' hearing. Finally, it is clear that our collaborators from ARL was able to adapt OSP to their own needs in this study, and achieved results for their research, that can now be more easily reproduced in the community. "Goldilocks" app is available as a part of our GitHub repository: https://github.com/nihospr01/OpenSpeechPlatform-UCSD.

6 DISCUSSION

We have successfully arrived at a solution which starts to bridge the "valley of death". The problem space is still quite vast and daunting. However, by adopting the minimum viable product approach to designing the system, we made the problem more tractable. The short development cycles with deliverable kept the project on track and helped obtain plenty of feedback in the early stages. In every cycle we were able to improve and innovate upon the previous system. This has brought us closer to bridging the gap. At the end of our third cycle we were able to evaluate the demo unit both in the lab and in the field.

The results show that the platform we have designed and developed allows algorithm developers to design real-time components without having to worry about the effects of users interacting with the system. At the same time the platform provides UX developers and audiologist a playground to design and develop different interfaces for hearing aid research. This allows the two worlds to co-exist almost in isolation of each other. However, when it comes to integrating the two worlds, the RT-MHA framework provides a simple yet powerful APIs for the task. In our experience with this software model, we have noticed a decrease in development cycle since developers can iterate in isolation from the rest of the system and thus the integration cycle is relatively quick. This needs to be further studied and quantified to understand the impact of this development model.

One of our major concerns while evaluating the portable device using ANSI3.22 test was that the high equivalent input noise would have prevent field studies from occurring until the issue with the codec was fixed. However, we were surprised at how the majority of the participants in the study rated the acoustic quality as good, even when comparing to their own hearing aid. One possible explanation is that the noise is bellow the perceivable range of most of the users. However, participant 1UA08 mentioned that the "1kHz tone was irritating" but still gave it a "Good" rating for the acoustical quality. So there must be another factor that needs to be explored.

7 CONCLUSION

In this paper, we describe how we designed and developed an opensource, modular, and extensible platform – Open Speech Platform – for hearing aid research. We present the design of both the hardware and software, with the evolution of the system from its conception till the current version through iterative improvements based on feedback from our collaborators. Next, we evaluate OSP's performance w.r.t both hardware and software, and also demonstrate its potential in adoption for research experiments and studies via the "Goldilocks" usability study involving human participants. The experience with OSP forms a basis for us to further develop and evaluate the system in the future. More information about the Open Speech Platform can be found at http://openspeechplatform.ucsd.edu/.

ACKNOWLEDGEMENTS

We would like to thank all of our colleagues that have contributed to the Open Speech Platform project and have made it the success it is currently. The author's would especially like to thank the rest of the Principal Investigators of this project Dr. Ganz Chockalingam, Prof. Patrick Mercier and Prof. Bhaskar Rao for all of their help and support on this project. Please visit http://openspeechplatform.ucsd.edu for the list of all of the people that worked with us on the Open Speech Platform.

We would like to thank Dr. Arthur Boothroyd's team at San Diego State University: Prof. Carol Mackersie, Paul Grand, Dr. Christine Kirsch, Dannielle Martin, Robert Novak, Shaelyn Painter, and Elena Shur for helping gather the very vital feedback from the users during their pilot field study.

We would like to thank the Wrethinking Foundation for their generous gift.

We thank the anonymous reviewers for the constructive comments.

This work is supported in part by the National Institute of Health, NIH/NIDCD grants R21DC015046, R33DC015046, and R01DC015436; the U.S. Army Research Laboratory Contract W911QX-16-C-0003; the NSF Graduate Research Fellowship DGE-1144086, and IIS-1838830; the CONIX Research Center, SRC JUMP; and the Qualcomm Institute.

REFERENCES

[1]. Boothroyd Arthur, Kirsch Christine, Mackersie Carol, Painter Shaelyn, and Garudadri Harinath. 2019. Usability assessment of a wearable speech-processing platform. The Journal of the Acoustical Society of America 146, 4 (2019), 2878–2878.

[2]. Boothroyd Arthur and Mackersie Carol. 2017. A "Goldilocks" approach to hearingaid self-fitting: User interactions. American journal of audiology 26, 3S (2017), 430–435. [PubMed: 29049625]

[3]. Doclo Simon, Gannot Sharon, Moonen Marc, Spriet Ann, Haykin Simon, and Liu KJ Ray. 2008. Acoustic beamforming for hearing aid applications. Handbook on array processing and sensor networks (2008), 269–302.

[4]. Doumler Timur. 2015. CppCon 2015: Timur Doumler "C++ in the Audio Industry" https://www.youtube.com/watch?v=boPEO2auJj4

[5]. Knowles Electronics. 2009. RVA-90020-NXX Datasheet

[6]. Knowles Electronics. 2014. RVA-90080-NXX Datasheet

[7]. Garudadri Harinath, Boothroyd Arthur, Lee Ching-Hua, Gadiyaram Swaroop, Bell Justyn, Sengupta Dhiman, Hamilton Sean, Krishna Chaithanya Vastare Rajesh Gupta, and Rao Bhaskar D. 2017. A realtime, open-source speech-processing platform for research in hearing loss compensation In 2017 51st Asilomar Conference on Signals, Systems, and Computers. IEEE, 1900–1904.

[8]. Herzke Tobias, Kayser Hendrik, Loshaj Frasher, Grimm Giso, and Hohmann Volker. 2017. Open signal processing software platform for hearing aid research (openMHA) In Proceedings of the Linux Audio Conference. CCRMA, Stanford University, California, US, 35–42.

[9]. Jorgensen Lindsey E. 2016. Verification and validation of hearing aids: Opportunity not an obstacle. Journal of otology 11, 2 (2016), 57–62. [PubMed: 29937811]

[10]. Kates James M. 2005. Principles of digital dynamic-range compression. Trends in amplification 9, 2 (2005), 45–76. [PubMed: 16012704]

[11]. Kates James M. 2008. Digital hearing aids Plural publishing.

[12]. Kates James M. 2019. Limitations of the Envelope Difference Index as a Metric for Nonlinear Distortion in Hearing Aids. Ear and hearing (2019).

[13]. Kayser Hendrik, Herzke Tobias, Maanen Paul, Pavlovic Chaslav, and Hohmann Volker. 2019. Open Master Hearing Aid (openMHA)—An integrated platform for hearing aid research. The Journal of the Acoustical Society of America 146, 4 (2019), 2879–2879.

[14]. Keidser Gitte, Dillon Harvey, Flax Matthew, Ching Teresa, and Brewer Scott. 2011. The NAL-NL2 prescription procedure. Audiology research 1, 1 (2011).

[15]. Kim Gibak, Lu Yang, Hu Yi, and Loizou Philipos C. 2009. An algorithm that improves speech intelligibility in noise for normal-hearing listeners. The Journal of the Acoustical Society of America 126, 3 (2009), 1486–1494. [PubMed: 19739761]

[16]. Kirchberger Martin J and Russo Frank A. 2015. Development of the adaptive music perception test. Ear and Hearing 36, 2 (2015), 217–228. [PubMed: 25350404]

[17]. Klasen Thomas J, Van den Bogaert Tim, Moonen Marc, and Wouters Jan. 2007. Binaural noise reduction algorithms for hearing aids that preserve interaural time delay cues. IEEE Transactions on Signal Processing 55, 4 (2007), 1579–1585.

[18]. Yoon kyu Sung Lingsheng Li, Blake Caitlin, Betz Josh, and Lin Frank R.. 2016. Association of Hearing Loss and Loneliness in Older Adults. Journal of Aging and Health 28, 6 (2016), 979–994. https://doi.org/10.1177/0898264315614570 arXiv:https://doi.org/10.1177/0898264315614570. [PubMed: 26597841]

[19]. Statistical Signal Processing Research Laboratory. 2020. Research and Development Platform - SSPRL - The University of Texas at Dallas Available at https://www.utdallas.edu/ssprl/hearing-aid-project/research-platform (2020/01/15).

[20]. Mackersie Carol, Boothroyd Arthur, and Lithgow Alexandra. 2019. A “Goldilocks” approach to hearing aid self-fitting: Ear-canal output and speech intelligibility index. Ear and hearing 40, 1 (2019), 107–115. [PubMed: 29894379]

[21]. Carol L Mackersie Arthur Boothroyd, and Garudadri Harinath. 2020. Hearing Aid Self-Adjustment: Effects of Formal Speech-Perception Test and Noise. Trends in hearing 24 (2020), 2331216520930545. [PubMed: 32552604]

[22]. Maidment David W, Ali Yasmin HK, and Ferguson Melanie A. 2019. Applying the COM-B model to assess the usability of smartphone-connected listening devices in adults with hearing loss. Journal of the American Academy of Audiology 30, 5 (2019), 417–430. [PubMed: 31044693]

[23]. Miller Roger L and Donahue Amy. 2014. Open Speech Signal Processing Platform Workshop https://www.nidcd.nih.gov/research/workshops/open-speech-signal-processing-platform/2014

[24]. Miller Roger L and Donahue Amy. 2014. Open Speech Signal Processing Platform Workshop

[25]. NIH. 2017. Hearing Loss and Hearing Aid Use (infographic) https://www.nidcd.nih.gov/shareable-images/infographic-hearing-loss-and-hearing-aid-use

[26]. Ohlenforst Barbara, Souza Pamela E, and MacDonald Ewen N. 2016. Exploring the relationship between working memory, compressor speed and background noise characteristics. Ear and hearing 37, 2 (2016), 137. [PubMed: 26517451]

[27]. World Health Organization et al. 2017. Global costs of unaddressed hearing loss and cost-effectiveness of interventions: a WHO report, 2017 World Health Organization.

[28]. Rallapalli Varsha, Anderson Melinda, Kates James, Balmert Lauren, Sirow Lynn, Arehart Kathryn, and Souza Pamela. 2019. Quantifying the Range of Signal Modification in Clinically Fit Hearing Aids. Ear and hearing (2019).

[29]. Russo Frank A and Pichora-Fuller M Kathleen. 2008. Tune in or tune out: age-related differences in listening to speech in music. Ear and hearing 29, 5 (2008), 746–760. [PubMed: 18596643]

[30]. Serrano Nicolas, Hernantes Josune, and Gallardo Gorka. 2013. Mobile web apps. IEEE software 30, 5 (2013), 22–27.

[31]. Souza Pamela, Arehart Kathryn, Schoof Tim, Anderson Melinda, Strori Dorina, and Balmert Lauren. 2019. Understanding Variability in Individual Response to Hearing Aid Signal Processing in Wearable Hearing Aids. Ear and hearing 40, 6 (2019), 1280–1292. [PubMed: 30998547]

[32]. Stone Michael A and Moore Brian CJ. 1999. Tolerable hearing aid delays. I. Estimation of limits imposed by the auditory path alone using simulated hearing losses. Ear and Hearing 20, 3 (1999), 182–192. [PubMed: 10386846]

[33]. Stone Michael A and Moore Brian CJ. 2002. Tolerable hearing aid delays. II. Estimation of limits imposed during speech production. Ear and Hearing 23, 4 (2002), 325–338. [PubMed: 12195175]

[34]. Stone Michael A and Moore Brian CJ. 2003. Tolerable hearing aid delays. III. Effects on speech production and perception of across-frequency variation in delay. Ear and Hearing 24, 2 (2003), 175–183. [PubMed: 12677113]

[35]. Stone Michael A and Moore Brian CJ. 2005. Tolerable hearing-aid delays: IV. Effects on subjective disturbance during speech production by hearing-impaired subjects. Ear and Hearing 26, 2 (2005), 225–235. [PubMed: 15809547]

[36]. Stone Michael A and Moore Brian CJ, Meisenbacher Katrin, and Derleth Ralph P. 2008. Tolerable hearing aid delays. V. Estimation of limits for open canal fittings. Ear and Hearing 29, 4 (2008), 601–617. [PubMed: 18469715]

[37]. Tahden Maike AS, Gieseler Anja, Meis Markus, Wagener Kirsten C, and Colonius Hans. 2018. What keeps older adults with hearing impairment from adopting hearing aids? Trends in hearing 22 (2018), 2331216518809737. [PubMed: 30451099]

[38]. Tympan. 2020. Tympan Available at https://shop.tympan.org (2020/01/15).

[39]. Van Vliet Dennis. 2010. Practicing audiology long-distance. The Hearing Journal 63, 2 (2010), 60.

[40]. Verma Lukeshwari, Kumar Sanju Himanshu, Scaria Bibina, Awasthi Mayank, Ravichandran Aparna, Kaki Ashritha, and Rathna Prakash Savalam Gnana. 2017. A comparative study on hearing aid benefits of digital hearing aid use (BTE) from six months to two years. International archives of otorhinolaryngology 21, 03 (2017), 224–231. [PubMed: 28680489]

[41]. Xanthopoulos Spyros and Xinogalos Stelios. 2013. A comparative analysis of cross-platform development approaches for mobile applications. In Proceedings of1 the 6th Balkan Conference in Informatics. ACM, 213–220.

CCS CONCEPTS

● **Applied computing** → **Consumer health**; • **Human-centered computing**; • **Hardware** → Printed circuit boards; • **Software and its engineering** → **Real-time systems software**; *Embedded software*; • **Computer systems organization** → *Sensors and actuators*; *Embedded hardware*;

Listing 1:

Template of a Library Module

```
class libModule{
public:
        ...
        /*@brief Setting libModule parameters*/
        void set_param(...){
              /* Create a new struct for the incoming param*/
              std::shared_ptr<libModule_param_t> next_param =
                           std::make_shared<libModule_param_t> ();
              std::atomic_store(&globalParam, next_param);
         }
         /*@brief Getting libModule parameters*/
         void get_param(...){
               /* Load the current global param structure atomically*/
               std::shared_ptr<libModule_param_t> localParam =
                            std::atomic_load(&globalParam);
               /* Return all of the parameters by reference*/
         }
         /*@brief Real-time processing inside libModule*/
         void process(...){
               /* Load the current global param structure atomically*/
               std::shared_ptr<libModule_param_t> localParam =
                            std::atomic_load(&globalParam);
               /* This is where the real-time code will go*/
         }
private:
         struct libModule_param_t {
               /* Set of Parameters */
         };
         /*The pointer to the global param structure*/
         std::shared_ptr<libModule_param_t> globalParam;
};
```

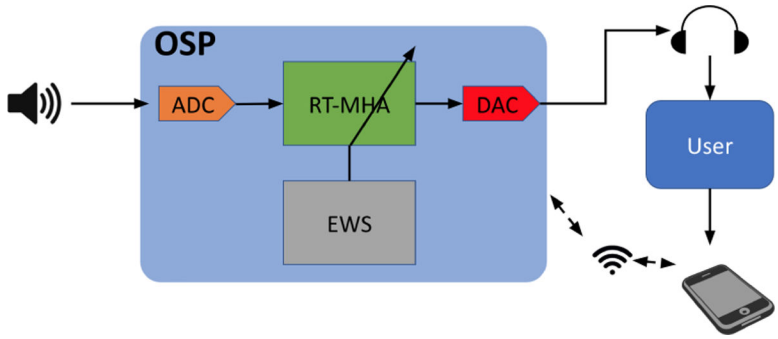

Figure 1:
Overview of the Open Speech Platform (OSP): OSP consists of both open-sourced hardware that meets the industry standards and software equipped with open, modular architecture and amenable user interfaces as well as functionality.

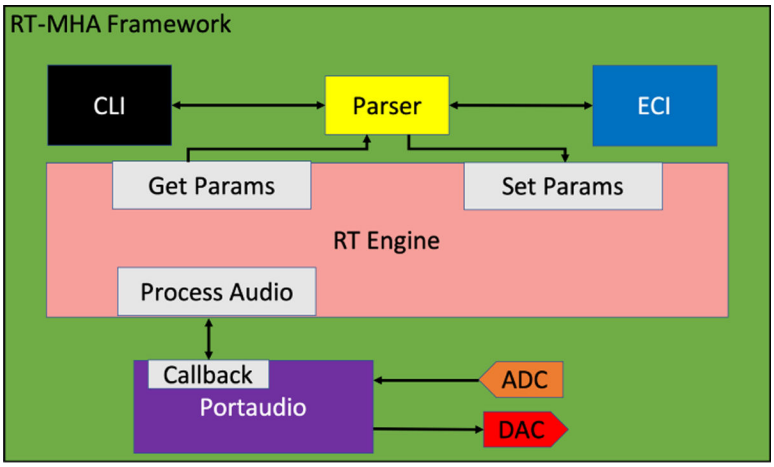

Figure 2:
Architecture of real-time master hearing aid.

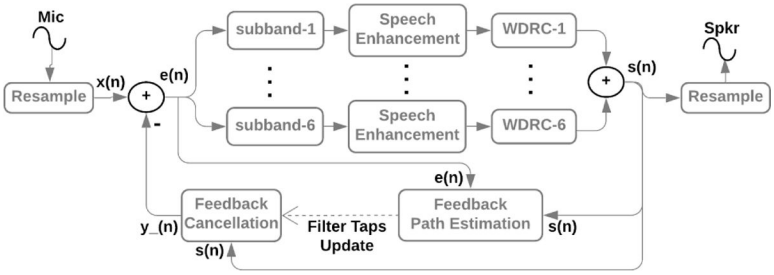

Figure 3:

Digital signal processing path for the reference RT-MHA implementation, comprised of audio resampling, sub-band filtering, speech enhancement, wide dynamic range compression, and feedback path estimation as well as feedback cancellation.

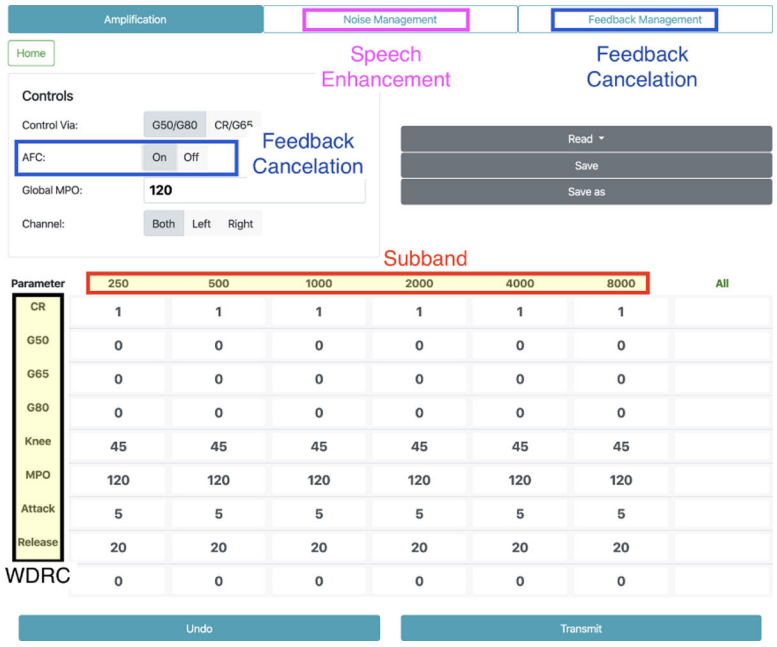

Figure 4:
Example graphical user interface for adjusting the reference design, mainly for researchers.

Figure 5:
The BTE-RIC hearing aid developed for both the lab-based system and the portable system.

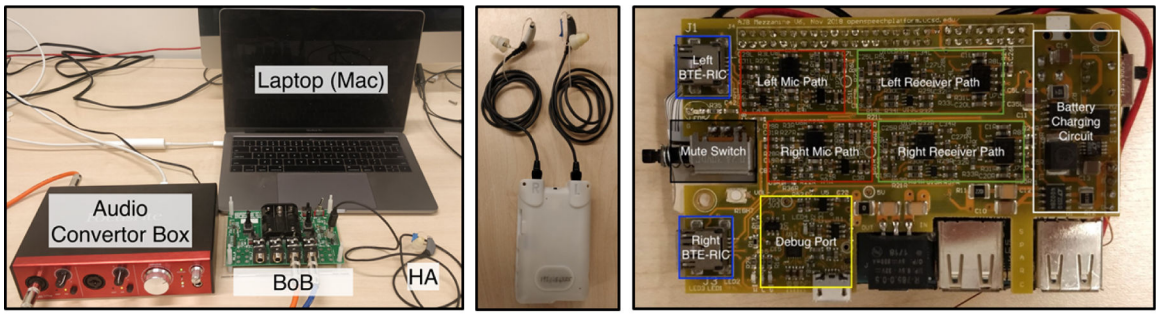

Figure 6:

a) Lab-based system consisting of a laptop (Mac), an audio converter box, a break out board, and a hearing aid. b) The portable system connected to two hearing aids. c) Annotated PCB view of the daughterboard that attaches to the the DragonBoard 410c SBC.

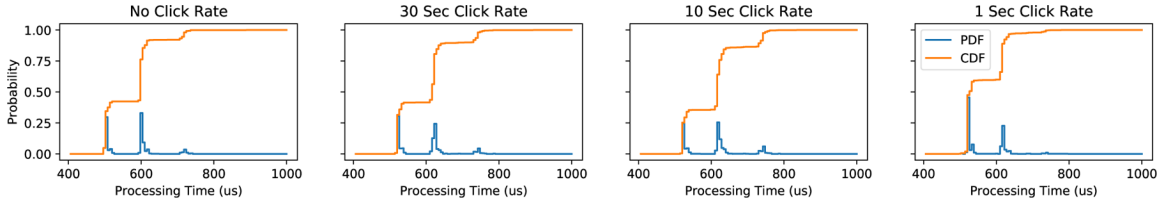

Figure 7:

The probability distribution (PDF) and cumulative distribution (CDF) of time required for the Real-Time Engine to process audio data packets over one-hour period under different request rates; it is capable to process data with no latency.

Figure 8:

Apparatus used for the SFP study. The arrows represent the information flow from the listener interface, via WiFi, to the Portable OSP to tune the WRDC.

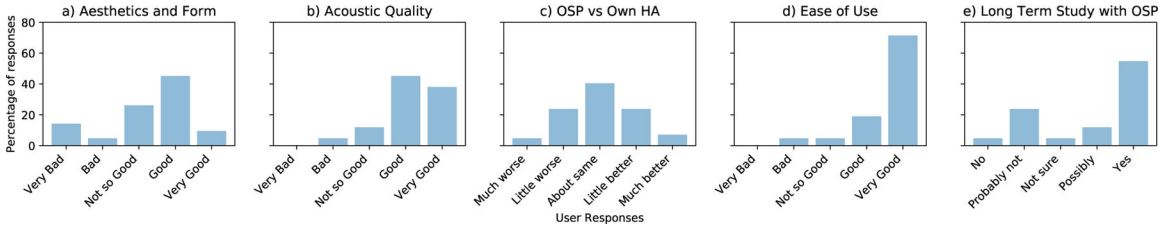

Figure 9:

Ratings (in histograms) by the participants in the usability study, regarding a) aesthetics and form of the portable OSP device; b) sound quality; c) sound quality of the portable OSP compared to their own hearing aids; d) level of ease-to-use of the web-app; and e) whether they are willing to participate in a research project with the portable OSP.

Table 1:

ANSI 3.22 test results for OSP system configurations measured by Audioscan Verifit 2, as compared to results from four commercial HAs.

Metric	*Units*	Commercial				Lab		Portable		ANSI
		A	**B**	**C**	**D**	**X**	**Y**	**X**	**Y**	**3.22**
Average Gain @ 60 dB	dB_{SPL}	40	40	25	35	40	40	35	39	–
Max OSPL90	dB_{SPL}	107	112	110	111	121	130	119	130	–
Average OSPL90	dB_{SPL}	106	109	108	106	112	126	111	126	–
Average Gain @ 50 dB	dB_{SPL}	37	39	25	35	35	41	35	40	–
Low Cutoff	*kHz*	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	≤ 0.2
High Cutoff	*kHz*	5	6	5	6.73	8	6.3	8	5	≥ 5
Equivalent Input Noise	dB_{SPL}	27	26	30	27	29	28	38	39	≤ 30
Distortion @ 500 Hz	*% THD*	1	1	0	0	2	1	1	3	≤ 3
Distortion @ 800 Hz	*% THD*	1	1	0	0	3	2	1	2	≤ 3
Distortion @ 1600 Hz	*% THD*	0	0	0	0	1	1	1	1	≤ 3

X: with high-bandwidth Knowles receiver [5]

Y: with high-power Knowles receiver [6]

Table 2:

Latency results for both the OSP systems and the RT-MHA reference design.

	Latency (ms)
Lab System (Mac)	5
Portable System	3.2
RT-MHA Reference	3.3
Lab + RT-MHA	8.3
Portable + RT-MHA	6.5