Requirements Document

Saving Your Ears and Sanity with Your Smartphone

Team Name	Dumbo	
Team Members	Josh Church	churchjo@onid.orst.edu
	Aidan Lance	lancea@onid.orst.edu
Client/Sponsor/Mentor	Sponsor name:	Virginia Balbi
	Organization:	Four Ears
	Phone:	541-250-0478
	Email:	Four4ears@gmail.com

Contents

Introduction to the Problem

Noise, welcome or not, is all around us. Although people have learned to adapt to its presence, sometimes there's just too much around us. By using special noise cancelling headphones, people have been able to retreat to a secluded, semi-noiseless world. Surprisingly, there hasn't been any mobile solutions for this problem, despite the advancements made in the mobile device markets. Since low-end noise cancelling headphones are rather expensive (for headphones), it would make sense, both logically and financially, if this functionality were adapted so that it would be able to run on smartphones.

Project Description

We will be creating a mobile application which will be able to get rid of unwanted ambient noise, thus creating an atmosphere more conducive for relaxation and concentration for the users of our product.

Requirements

I. Essential Elements:
   1. The application shall be able to actively cancel low frequencies
   2. The application shall be able to activate its noise cancellation mode upon user request, given that the mode is not active at the time of the request
   3. The application shall be able to deactivate its noise cancellation mode upon user request, given that the mode is active at the time of the request

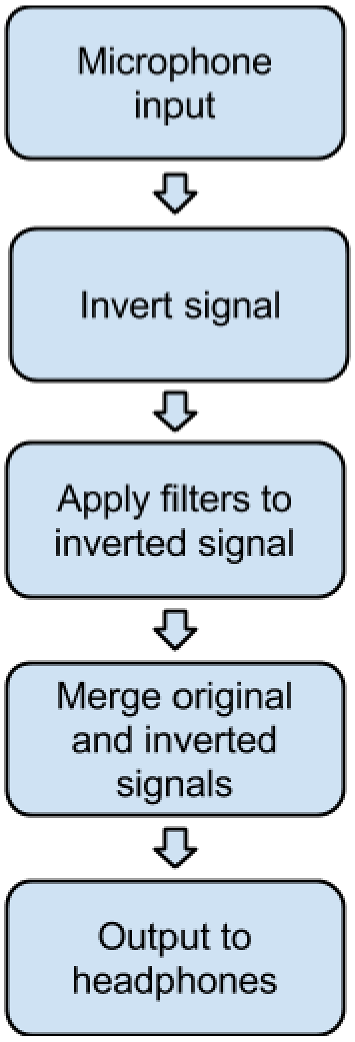
II. Desired Elements:
   1. The application shall be able to actively cancel higher frequencies
   2. The application shall be able to perform the following actions given that the low frequency cancellation mode is active:
      a. The application shall be able to activate its high frequency cancellation mode upon user request, given that the mode is not active at the time of the request
      b. The application shall be able to deactivate its high frequency cancellation mode upon user request, given that the mode is active at the time of the request
      c. The application shall keep the low frequency cancellation mode active while the high frequency cancellation mode is active
   3. The application shall only activate the high frequency cancellation mode if the low frequency cancellation mode is active
   4. The application shall be able to process an incoming signal with no more than a 500ms delay between receiving a signal and outputting the modified signal to the user.

III. Non-Essential Elements:
   1. The application shall be able to filter different frequency ranges, allowing for customization of the active frequency cancellation.

Versions

Version 0.01	Bare-bones application focused on the implementation of the low frequency noise cancellation system. This will be nothing fancy, and will thusly have a basic user interface (for example, a single toggle button to turn noise cancellation on and off).
Version 0.25	Address issues from the previous release. Improvements to low frequency noise cancellation systems will be made. Reduce delay resulting from signal processing.
Version 0.50	Address issues from the previous release. Initial high frequency cancellation will be introduced. Additional features will also be added, and will be attached to corresponding user interface controls.
Version 0.75	Address issues from previous release. Improve high frequency noise cancellation. Reduce delay resulting from signal processing.
Version 1.00	Address issues from previous release. Improve efficiency and speed of noise cancellation algorithms. Improve user interface by removing unneeded controls. The interface's layout will also be reorganized to ensure maximal aesthetic appeal and usability.
Version 2.00	Address issues from previous release. Optimize user interface for speed and efficiency.

Design


Microphone input
Invert signal
Apply filters to inverted signal
Merge original and inverted signals
Output to headphones

There may also be additional FFT algorithms which transform our signal after receiving and prior to transmitting our respective input and output signals. However, since the input is already in the frequency domain, this may add additional and unnecessary processing time which may result in an audible delay effect. We will continue to research whether this is, in fact, a necessary component.

Specific Tasks to be Undertaken

I. Microphone input
   1. Create efficient function to get microphone input from a mobile device

II. Headphone Output
   2. Output mono audio to a mobile device's audio out port

III. Noise cancellation
   1. Create a Fast Fourier Transform (FFT) algorithm to quickly process input from a microphone (or some other source of signal input)
   2. Reverse the signal obtained from the FFT algorithm and merge it with the original signal.
   3. Output the signal to the user through the headphone input of the mobile device

IV. User interface
   1. Connect a UI toggle button to the noise cancellation process so that the user can turn the cancellation feature on or off

Risk Assessment

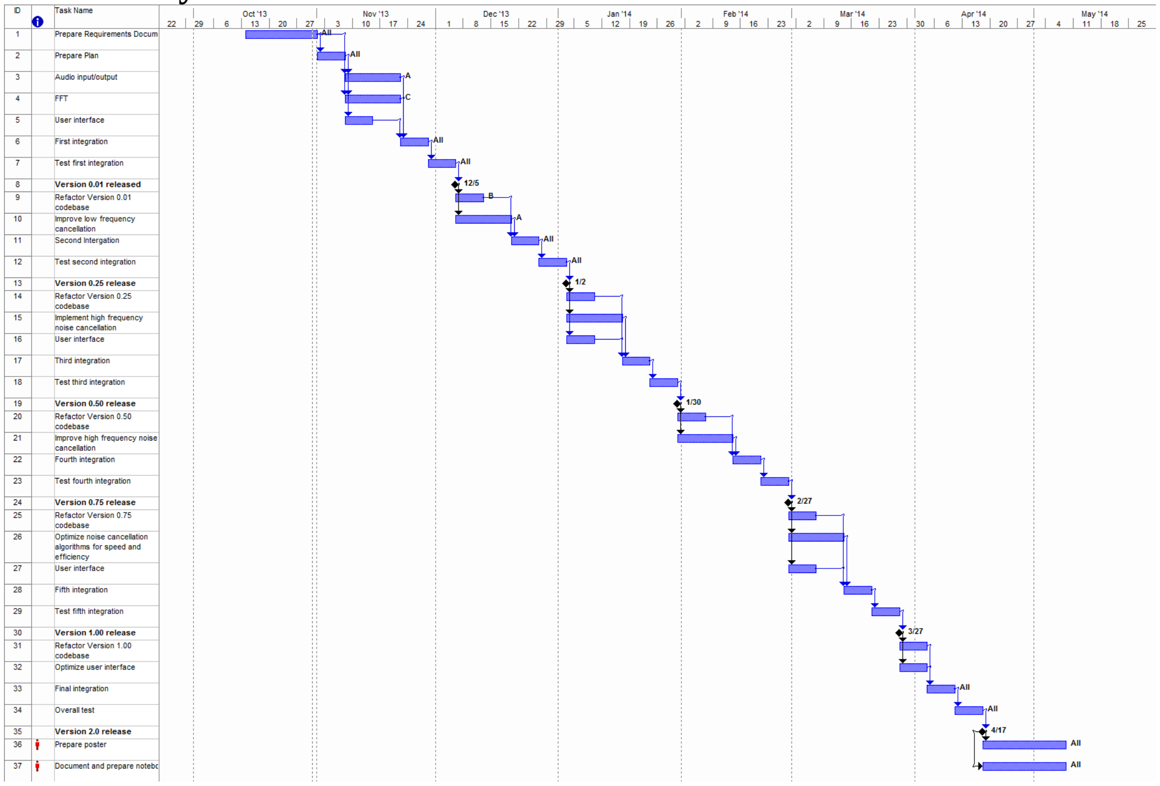
Lack of communication between the developers and the client can cause our project to slow to a halt. We have prepared for this by scheduling regular bi-monthly meetings. We will also schedule a day where code reviews will be conducted by all developers. In scheduling these, we will be able to check each other's pull requests before integrating any code into our final product. Communication issues are fairly obvious to see. For example, if contact cannot be made with a team member or the client for a significant amount of time, then there is obviously a communication issue. If we do encounter any issues resulting from significant communication mishaps, then we will contact our TA and Dr. Bailey immediately. Smaller communication problems will be addressed by our team, and if we are unable to find a solution, then we will consult our client, or the TA and Dr. Bailey.

Porting to iOS from Android is going to be a very risky maneuver. We can lose overall efficiency and even total functionality. We will prepare for this by having all of our code and algorithms heavily documented. That way, if we needed to rewrite our application in another language, it would be a trivial task for us to translate the entirety of the application. If we are having trouble porting functionality to a different platform, the developer responsible for porting the functionality will consult the developer who wrote the original code in order to gain a better understanding of what actions are being performed by the code. If the original developer is unable to provide sufficient information, then the porting developer will seek out additional resources (club members, internet forums, books, etc.) which will be beneficial in enlightening the developer on the subtleties of the original implementation.

Testing

Our testing philosophy is fairly simple. Unit tests will be written for functions in order to test each function’s ability to handle the various inputs, and ensure that the functional outputs are expected and appropriate for the tested situations. Unit tests will be written as the functions are being written. Additional reliance will be placed on integration testing to ensure that older code from previous versions are still functional with any additional new code changes. Integration testing will be completed by testing through our “public” API when possible. We will also utilize black box testing to test a majority of our final product code. We will also partially rely on feedback from our client with regards to our releases. Her feedback will help us determine where we need to focus our attention for the next release.

Preliminary Timetable



ID		Task Name
1		Prepare Requirements Docum
2		Prepare Plan
3		Audio input/output
4		FFT
5		User interface
6		First integration
7		Test first integration
8		**Version 0.01 released**
9		Refactor Version 0.01 codebase
10		Improve low frequency cancellation
11		Second Intergation
12		Test second integration
13		**Version 0.25 release**
14		Refactor Version 0.25 codebase
15		Implement high frequency noise cancellation
16		User interface
17		Third integration
18		Test third integration
19		**Version 0.50 release**
20		Refactor Version 0.50 codebase
21		Improve high frequency noise cancellation
22		Fourth integration
23		Test fourth integration
24		**Version 0.75 release**
25		Refactor Version 0.75 codebase
26		Optimize noise cancellation algorithms for speed and efficiency
27		User interface
28		Fifth integration
29		Test fifth integration
30		**Version 1.00 release**
31		Refactor Version 1.00 codebase
32		Optimize user interface
33		Final integration
34		Overall test
35		**Version 2.0 release**
36		Prepare poster
37		Document and prepare notebc

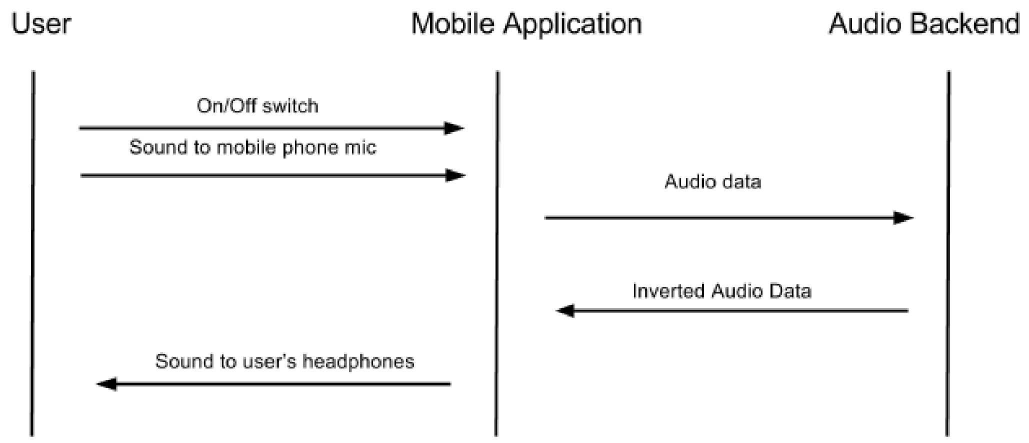
Team Member Roles

Considering we are only a two person team, our workload will frequently overlap. We will be creating an audio backend, and then later implementing it on Android and iOS. Josh will be primarily working on the Android Implementation, and Aidan will be primarily working on the iOS implementation. We will be reviewing each other's code on GitHub using pull requests.

Integration Plan

A series of integration tests will ensure that the components that we make will work with the rest of the project. Integration of the code into the codebase will be done using GitHub, and the integration checks will be performed by the team member who wasn't responsible for writing the code in question.

Dataflow Sequence Diagram



User Interface Requirements

1. The application shall have toggle buttons to activate or deactivate supported active noise cancellation modes
2. The application shall disable the high frequency cancellation activation button if low frequency cancellation is not active
3. The application shall disable the low frequency deactivation button if the high frequency cancellation is active

References

Baldwin, Richard. "Adaptive Noise Cancellation Using Java." *Developer.com*. N.P., 18 April 2006. Web. 31 Oct. 2013.

Glossary

FFT	The **F**ast **F**urrier **T**ransform is an algorithm capable of rapidly converting time or space to frequency, or vice versa.
GitHub	A hosting service for programmers which utilizes the Git revision control system.
iOS	The operating system used on common Apple mobile devices (such as the iPhone, and the iPod Touch).
OS	An **O**perating **S**ystem manages and provides an interface for programs to a computer's resources (memory, processor, and other hardware commonly seen on computing devices).

Signatures

____

____

____