CS Capstone Design Document

April 19, 2019

Audio Extravaganza

Prepared for

Oregon State University

Kirsten Winters

Prepared by

Group26

Audio Extravaganza

Martin Barker

Devon Cash

Alexander Niebur

Mason Sidebottom

Ben Windheim

Abstract

The Audio Extravaganza project aims to build a modulation and looping pedal that is affordable, intuitive, and effective. This document lays out the design for the pedal. Starting with an overview of the projects scope, purpose and intended audience. Next, the conceptual model for the software design will be covered. Then viewpoints used to define our design will then be covered. Finally, the design itself from the viewpoints listed will be defined.

CONTENTS

1 TABLE OF CHANGES

Section	Original	New
Overview	• Section 2.1 indicated a combination of SuperCollider and Puredata would be used for development	• Only SuperCollider used for implementation, removed references to Puredata
Design Description	• Section 4.5.4 indicated interface design would be iterated on using user testing	• Language referencing user testing was removed
Design Viewpoints	• Section 5.2.1.1 described Looper and Audio Modulation systems as separate modes to switch between	• Removed use case dependent on this design construct
	• Section 5.3 described Looping system and Audio Modulation system as mutually exclusive	• Diagram updated to reflect current design
	• Section 5.5 had an outdated image of our proposed design	• Image replaced with current design
	• Section 5.6.2 indicated that audio effects were loaded just-in-time as opposed to on program start	• Sequence diagram updated to reflect changes to system design

2 Overview

2.1 Scope

The Audio Extravaganza project is designed to bring an affordable audio effects solution to consumers at a competitive price. Our final product will be an embedded system with a variety of audio effects, I/O ports for audio signals, knobs to control the audio effects, and a metallic frame to house the system. The audio effects will be programmed and run in the programming environments Supercollider on the BeagleBoard X15.

2.2 Purpose

This documentation is designed to describe the goals for the Audio Extravaganza project in a way that is easy to understand and maximizes coverage of the project objectives through observation of different viewpoints. Additionally, this document will allow our client to determine if our objectives correspond to the expectation set for the project.

2.3 Intended Audience

The following document is intended to be read by multiple parties:

- Kirsten Winters, our project client for review of project goals, expectations, and requirements.
- Kevin McGrath, for project understanding and grading purposes.
- Members of the Audio Extravaganza team, for use during the implementation portion of development alongside the case where a dispute about design decisions require reviewing our design choices.
- Members of the testing team and end-users, for understanding about how to use our final product and for finding any issues that could arise within our design document.

2.4 Conformance

The following document shall conform to the standards and specifications set by our client, and in the case that any specification is not met, we shall modify our design to accommodate. In addition, the requirements document explains the manufacturing quality for which the implementation needs to conform to. Said document will be a reference for validation of implementation results.

3 Conceptual model for software design descriptions

This section presents the conceptual model for which the Software Design Descriptions (further referred to as SDD) will be presented. The conceptual model will include basic terminology and concepts for the SDD, the surrounding concepts and context, and the stakeholders interested.

3.1 Software design in context

In the Audio Extravaganza project, the design method will be a blend of a variety of different approaches to software development. The overarching system will be module-based for creating and loading our separate effects/subsystems, which we will refer to as patches to avoid ambiguity of terminology when dissecting audio effects. Whether these patches are running concurrently (such as the combination of a modulation effect and a looper) or are being cycled through one at a time, the patches will need to maintain autonomy and distinction justifying the need for a load-in, load-out approach. However, the individual patches will be enveloped in an object-oriented design to encourage protection and separation for the control interface. This will help isolate potential issues with development and create for clear and readable logic for a strong and stable development cycle within the team.

3.2 Software design descriptions within the life cycle

3.2.1 Influences on SDD Preparation

The description and requirements for the Audio Extravaganza project software is clearly defined in the culmination of each team member's requirements and technical documents. The need for the previously mentioned design is based specifically on the results of these documents.

3.2.2 Influences on Software Life Cycle Products

The final software product of the Audio Extravaganza project is certainly highly dependent on the design choices list here in the SDD conceptual model as well as the entirety of this document. With respect to the SDD conceptual model, this design model influences the entirety of the software implementation phase, which we will complete before integration into hardware. The development cycle will consistently reference this model of implementation for the actualized implementation, and the final product's software will directly reflect the blended design decisions laid out before. In addition to design and implementation, test design and execution will be developed based on the design model constructed here.

3.2.3 Design Verification and Design Role in Validation

After documentation, rough test cases can be constructed to ensure the accuracy and validity of the design and implementation models constructed here, in the technical documents, and in the requirements documents. While many of the targeted goals and metrics for evaluation are fairly subjective, test cases before even beginning implementation can drive development in a way that suits the requirements for the project while adhering to the blended software development description design we have built. The range of design concerns present in this document can be addressed in a test-driven development scenario in which the metrics for success are defined before the implementation has begun or finished.

4 DESIGN DESCRIPTION INFORMATION CONTENT

4.1 Introduction

The following sections describe what viewpoints will be used and how they will be used to design the product. The information provided can be used to understand the goals of the design. The information can be found in the following sections: SDD identification, design stakeholders, design views, design viewpoints, design elements, design rationale, and design languages.

4.2 SDD identification

This document describes the system design for the Audio Extravaganza capstone project device following the standards described in *IEEE 1016-2009*.

4.3 Design stakeholders and their concerns

The stakeholders involved in the Audio Extravaganza project are the members of the development team and our client, Dr. Kirsten Winters. The stakeholders' concerns are listed below:

- The project is to create a device that can apply audio effects to an input from an instrument or microphone.

- The final device should be suitable for live music performance.
- The device's interface should be easy to use and accessible to users unfamiliar with other audio effects devices.
- The device's audio effects systems should be powerful enough to be useful to experienced audio effects users.
- The final price of the device should be affordable to improve its accessibility to novice users.

4.4 Design views

The view used to design the Audio Extravaganza capstone project device are:

- Context
- Composition
- Dependency
- Interface
- Interaction
- Resource

4.5 Design viewpoints

4.5.1 Context viewpoint

This viewpoint is concerned with how users and/or stakeholders will interact with the system in reference to a specific context. The elements used in this viewpoint will be *actors*, *relationships*, and *constraints*. In this case, our actors will be musicians, and people with extensive musical hardware experience, as well as novices and live performers. The relationships will be how much experience a user has with musical hardware such as effects pedals, as well as experience with music in general, as we would want our product to be intuitive enough for non-technical musicians to use. Constraints will be how aware our subjects and stakeholders are with effects pedal standards, such as using a large foot pedal input to trigger loops, and dials to adjust sound. Entities described in the context viewpoint will be done via UML use cases. [1]

4.5.2 Composition viewpoint

This viewpoint describes the the role and structure of system elements. The elements referenced in this viewpoint will be *entities*, *relationships*, and *attributes*. For this project, the structure of our system elements will be mainly divided up into software and hardware. With software programming including creating modifiable effects capable of being transferred to our hardware, which will need to be securely housed, reliably powered, and user friendly. These viewpoint will be described through UML component diagrams to clearly demonstrate the hierarchies and structure.

4.5.3 Dependency viewpoint

This viewpoint describes how entities are connected and how they should share resources. Hardware and software entities will share power, audio, and input/output resources. Therefore computationally intensity should be considered when designing the audio effects and software for our board, as we don't want to end up in a position where throttled computational power limits our progress. Power draw for our operating system, wireless external interface transmission, and input/output interaction will need to operate together seamlessly. Utilizing our board's audio card will also be important to ensure that resources are allocated in a fair and efficient way, such that a simple low pass filter is not being repeated for each effect. The software effects code should be modularized whenever possible. The notation used to describe this viewpoint will be UML package diagrams.

4.5.4 Interface viewpoint

The interface viewpoint describes how users can expect to interact with the object. This will be described with a mock-up of the physical interface.

4.5.5 Interaction viewpoint

The interaction viewpoint describes how and why entities should work work together. Actions which occur at the hardware level (adjustment of knobs, stomping of the pedal) should take precedent over what the software does. Since supporting the opposite separation of privilege would cause instances of frustration for users when their physical manipulation of a device is not regarded. This will be described with a UML interaction diagram.

4.5.6 Resource viewpoint

The resource viewpoint describes elements that are part of the system but external to the design. These components will be listed below:

- A small LCD screen.
- Two 1/4 inch female to 1/8 inch male audio adapter.
- An on/off switch.
- A foot pedal.
- An options dial.
- An intensity dial.

4.6 Design elements

4.6.1 File system

4.6.1.1 Name: File System

4.6.1.2 Type: Subsystem

4.6.1.3 Purpose: This element defines the organization of all files to be used by the overall system.

4.6.2 Looping system

4.6.2.1 Name: Looping system

4.6.2.2 Type: Subsystem

4.6.2.3 Purpose: This element defines the procedures that will record and playback audio.

4.6.3 Physical Interface

4.6.3.1 Name: Physical Interface

4.6.3.2 Type: Subsystem

4.6.3.3 Purpose: This element defines the physical actions required to use software systems and the hardware elements to display current system status.

4.6.4 Audio Modulation System

4.6.4.1 Name: Audio Modulation System

4.6.4.2 Type: Subsystem

4.6.4.3 Purpose: This element defines the procedures that will be used to apply audio effects to incoming signals.

4.6.5 Audio Effect Library

4.6.5.1 Name: Audio Effect Library

4.6.5.2 Type: Module

4.6.5.3 Purpose: This element defines the set of audio effects that will be included with the system.

4.7 Design rationale

The rationale behind the design of this system is to define a collection of modules that can work either independently or together. By splitting each major functionality into its own subsystem or module, we decrease the coupling, allowing for easier maintainability. Furthermore, the design demonstrates robust solutions that increase usability.

4.8 Design languages

The primary design language used is *Unified Modeling Language*, more commonly referred to as UML. In the case the the entity being described does not follow UML standards, a key and description of notation will be provided.

5 DESIGN VIEWPOINTS

5.1 Introduction

This chapter describes the design viewpoints relevant to this project in detail. Each section will address design concerns related to the design views denoted in section 4.4. Seven design viewpoints will be described:

- Context viewpoint
- Composition viewpoint
- Dependency viewpoint
- Interface viewpoint
- Interaction viewpoint
- Resource viewpoint

5.2 Context Viewpoint

5.2.0.4 Start recording audio: The user shall be able to record audio to playback later.

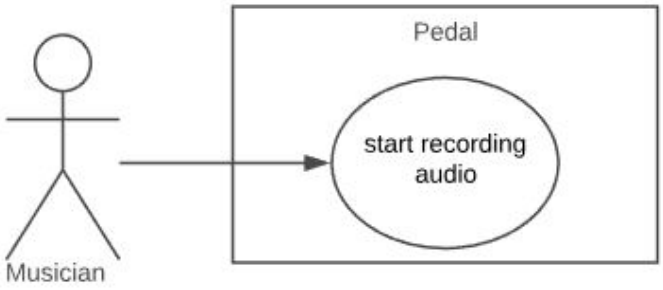

Fig. 1. Start recording audio use case

Use Case	Start recording audio
Use Case Number	2
Summary	Musician can start recording audio.
Actor	Musician
Trigger	Hardware input: pedal toggle
Pre-Conditions	The pedal employ the looping subsystem. The looping subsystem must not already be recording.
Post-Conditions	The looping subsystem will be recording incoming audio.
Assumptions	The user has audio input plugged in.

TABLE 1
Use case: Start recording audio

5.2.0.5 Stop recording audio: The user shall be able to record audio to playback later.

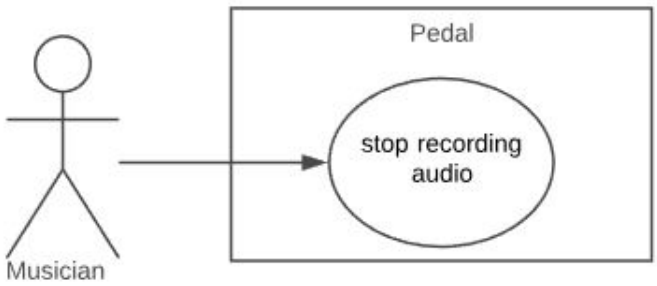

Fig. 2. Stop recording audio use case

Use Case	Stop recording audio
Use Case Number	3
Summary	Musician can stop recording audio.
Actor	Musician
Trigger	Hardware input: pedal toggle
Pre-Conditions	The pedal must be in the looping subsystem. The looping subsystem must be recording audio.
Post-Conditions	The looping subsystem will cache the recorded audio.
Assumptions	The user has audio input plugged in.

TABLE 2
Use case: Stop recording audio

5.2.0.6 Start audio playback: The user shall be able to playback recorded audio.

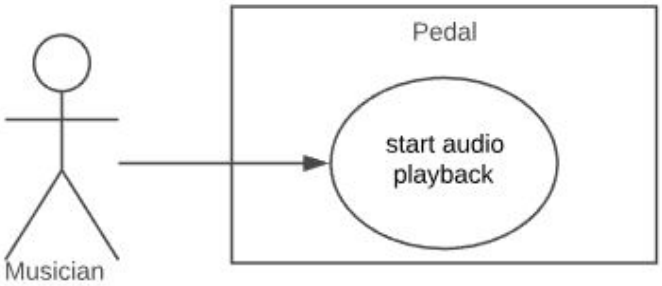

Fig. 3. Start audio playback use case

Use Case	Start audio playback
Use Case Number	4
Summary	Musician can start playing back audio.
Actor	Musician
Trigger	Hardware input: pedal toggle
Pre-Conditions	The pedal must be in the looping subsystem. The looping subsystem must have audio cached.
Post-Conditions	The looping subsystem will begin audio playback.
Assumptions	None.

TABLE 3
Use case: Start audio playback

5.2.0.7 Stop audio playback: The user shall be able to stop the playback recorded audio.

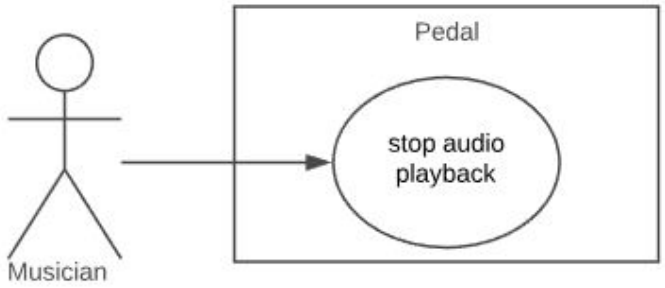

Fig. 4. Stop audio playback use case

Use Case	Stop audio playback
Use Case Number	5
Summary	Musician can stop audio playback.
Actor	Musician
Trigger	Hardware input: pedal toggle
Pre-Conditions	The pedal must be in the looping subsystem. The looping subsystem must have audio cached. The looping subsystem must be currently playing audio back
Post-Conditions	The looping subsystem will stop audio playback.
Assumptions	None.

TABLE 4
Use case: Stop audio playback

5.2.0.8 Select effect: The user shall be able to select an effect.

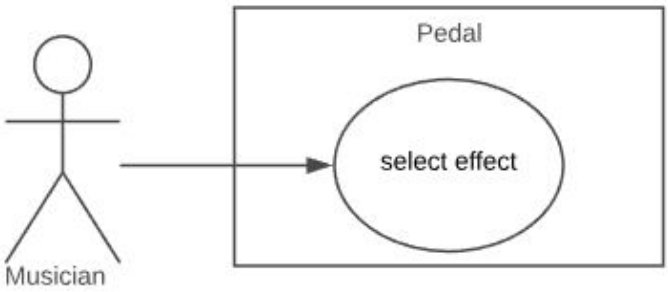

Fig. 5. Select effect use case

Use Case	Select effect
Use Case Number	6
Summary	Musician can select an effect to use on the pedal.
Actor	Musician
Trigger	Hardware input: selector
Pre-Conditions	The pedal must be in the modulation subsystem. The audio effect library must have effects.
Post-Conditions	The modulation subsystem will load an effect to a cache.
Assumptions	None.

TABLE 5
Use case: Select effect

5.2.0.9 Start effect: The user shall be able to start using an effect.

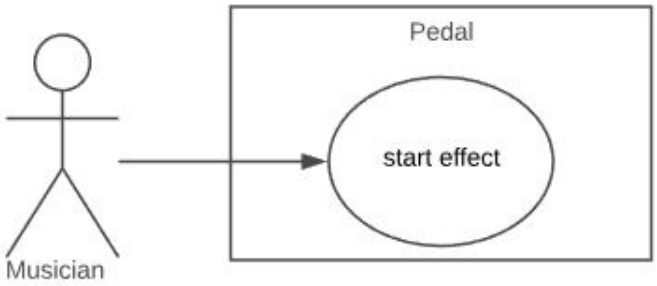

Fig. 6. Start effect use case

Use Case	Start effect
Use Case Number	7
Summary	Musician can start using an effect to modulate incoming audio.
Actor	Musician
Trigger	Hardware input: pedal toggle
Pre-Conditions	The pedal must be in the modulation subsystem. The corresponding pedal toggle must have an effect cached.
Post-Conditions	The modulation subsystem will start modulating incoming audio.
Assumptions	An incoming audio signal exists.

TABLE 6
Use case: Start effect

5.2.0.10 Stop effect: The user shall be able to stop using an effect.

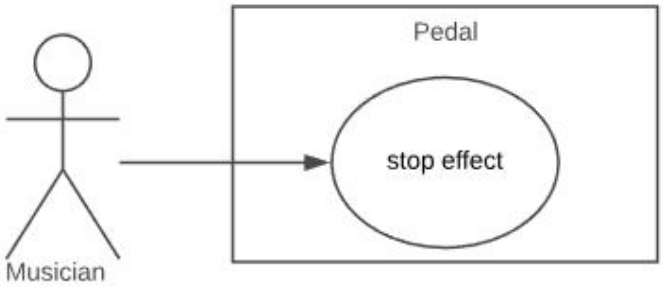

Fig. 7. Stop effect use case

Use Case	Stop effect
Use Case Number	8
Summary	Musician can stop using an effect to modulate incoming audio.
Actor	Musician
Trigger	Hardware input: pedal toggle
Pre-Conditions	The pedal must be in the modulation subsystem. An effect must be running.
Post-Conditions	The modulation subsystem will stop modulating incoming audio.
Assumptions	An incoming audio signal exists.

TABLE 7
Use case: Stop effect

5.3 Composition viewpoint

5.3.1 UML component diagram

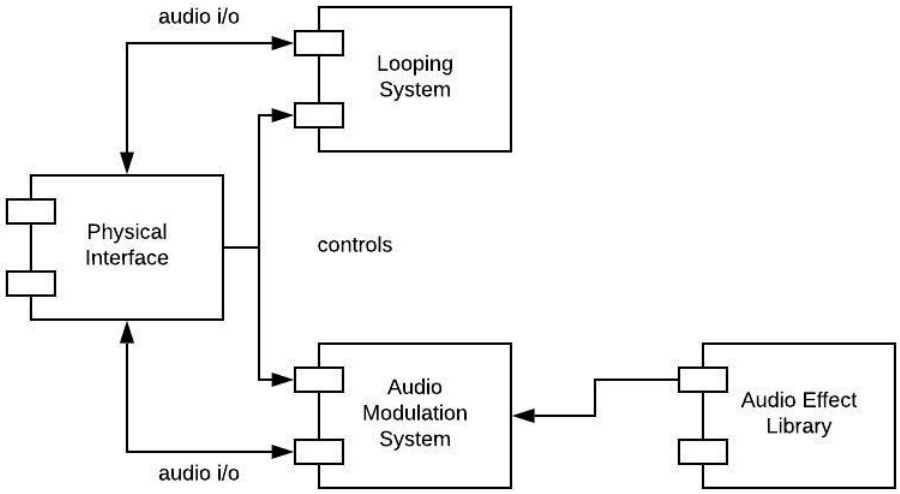

Fig. 8. Component diagram

5.3.2 Function attribute

The composition of the above entities creates the entire system to allow musicians to modulate incoming audio, or record and playback audio.

5.3.3 Subordinates attribute

The collection of the following entities work together to construct the working modulation/looping pedal: physical interface, looping system, audio modulation system, file system, and audio effect library.

5.4 Dependency viewpoint

5.4.1 UML dependency diagram

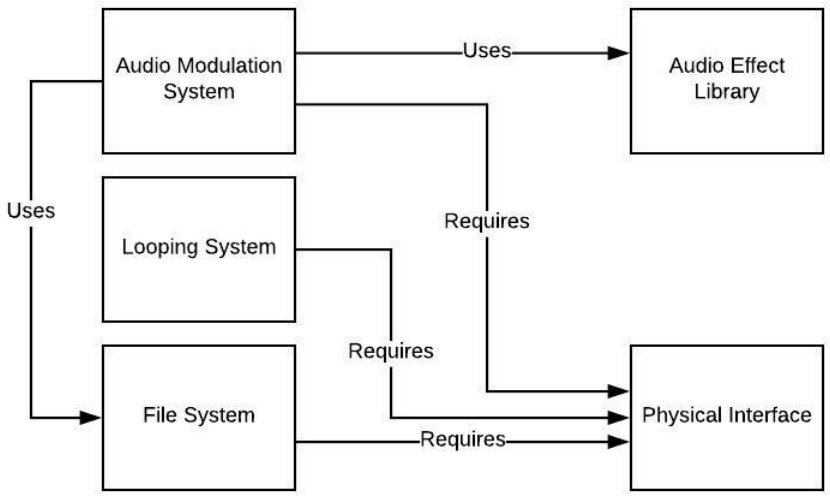

Fig. 9. Dependency diagram

5.4.2 Dependencies attribute

Figure 9 shows the development dependencies for each subsystem. The first subsystems that can be designed are the Audio Modulation and Looping subsystems. The Looping subsystem design is independent from other components and can be developed at any time before the Physical Interface. The design of the Audio Modulation Subsystem is used to inform the Audio Effect Library and the File System designs. Once the software subsystems (Audio Modulation System, Looping System, and File System) are designed, work can begin on the Physical Interface design to create an interface that will adequately and efficiently control all three software subsystems.

5.5 Interface viewpoint

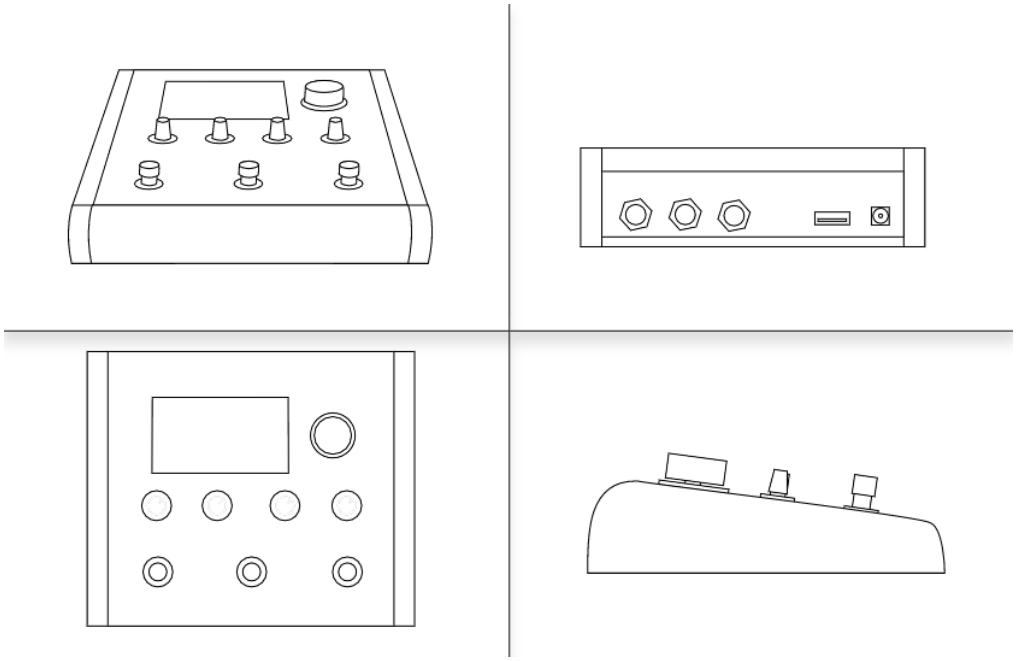

Fig. 10. Design for the final product

5.6 Interaction viewpoint

5.6.1 Looping System UML sequence diagram

Figure 11 shows the intended interaction of the Hardware Interface and Looping Subsystem. Physical controls on the Hardware Interface correspond directly to the functionality of the Looping system. Creating a one to one relationship between the physical controls on the hardware and the software functionality they control should facilitate intuitive control while affecting the Looping System.

5.6.2 Audio Modulation System UML sequence diagram

Figure 12 illustrates the intended interaction between the Hardware Interface and the Audio Modulation System.

5.7 Resource viewpoint

The resources that may be present in the product, but are not part of the design are:

- External display device.
- Open source libraries.
- Any behavior related to stretch goals.

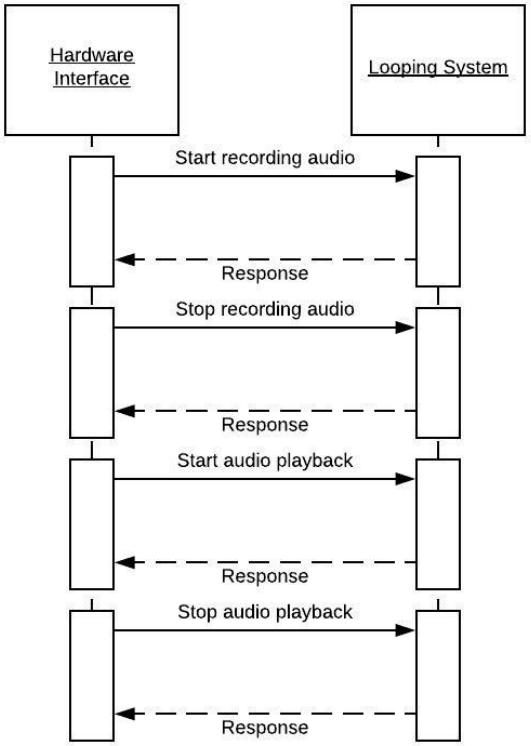

Fig. 11. Looping System sequence diagram

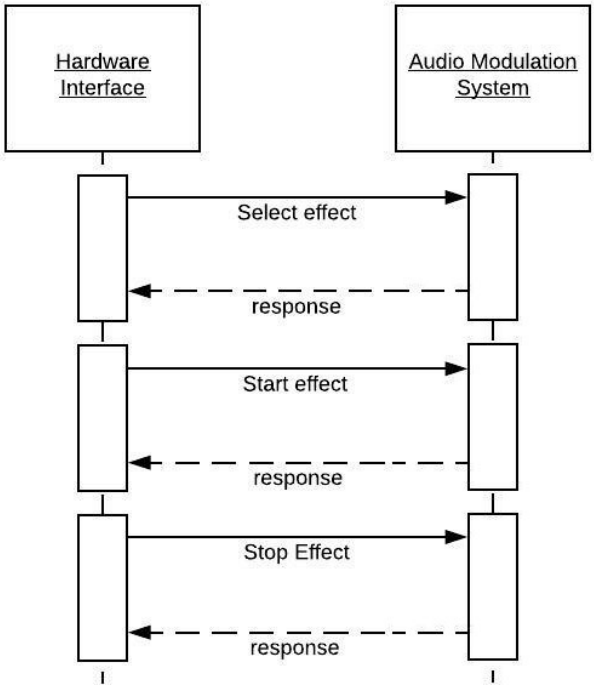

Fig. 12. Audio Modulation System sequence diagram

REFERENCES

[1] *IEEE STD 1016-2009*. IEEE, 2009.