**Lab 8 Report**

Manav Mandhani (mm58926)

Cyril Yannick Engmann (cye78)

A) Objectives

**Requirements document**

1. Overview

 1.1. Objectives: Why are we doing this project? What is the purpose?

The objectives of this project are to design, build and test a DJ Mixer to create songs. The operator will be able to create a multi layered song. Your goal is to create some dope beats.

 1.2. Roles and Responsibilities: Who will do what?  Who are the clients?

EE445L students are the engineers and the TA is the client. Students are allowed to divide responsibilities of the project however they wish, but, at the time of demonstration, both students are expected to understand all aspects of the design.

 1.3. Interactions with Existing Systems: How will it fit in?

The system will use the LM3S811 board, a soldered PCB board, and an audio jack. It will be powered using an external battery.

2. Function Description

 2.1. Functionality: What will the system do precisely?

The system will begin by playing a simple continuous n-second beat loop. When the operator presses one of the three instrument buttons a sound will be made and saved into that continuous loop, creating a song of 2 layers. That sound will repeat at the same time every n-seconds on top of the initial sound loop. The operator can use any of the three buttons to add a different noise and save it on the song loop. By mixing and matching different noises the operator can create a song. A fourth button will be used to clear all added notes to the simple drum beat.

An LCD will show a sound histrogram graphic.

There must be a C data structure to hold the music. There must be a music driver that plays songs. Although you will be playing only one song, the song data itself will be stored in a separate place and be easy to change. The player runs in the background using interrupts. The foreground (main) initializes the player, then executes **for(;;){}** do nothing loop. If you wish to include OLED output, this output should occur in the foreground.

 2.4. Performance: Define the measures and describe how they will be determined.

The system will be judged by three qualitative measures. First, the software modules must be easy to understand and well-organized. Second, the system must employ an abstract data structures to hold the sound and the music. There should be a clear transition when notes are added to the song. Backward jumps in the ISR are not allowed. Waiting for SSI output to complete is an acceptable backwards jump. Third, all software will be judged according to style guidelines. Software must follow the style described in Section 3.3 of the book. There are three quantitative measures. First, the SNR of the DAC output of a sine wave should be measured. Second, the maximum time to run one instance of the ISR will be recorded. Third, you will measure power supply current to run the system. There is no particular need to optimize any of these quantitative measures in this system.

 2.5. Usability: Describe the interfaces. Be quantitative if possible.

There will be four switch inputs. Each input will correspond to an instrument that will add another layer to the existing song. A fifth switch input will clear all layers to the original song. The DAC will be interfaced to a 32-ohm speaker. There will be an LCD.

3. Deliverables

 3.1. Reports: How will the system be described?

A lab report described below is due by the due date listed in the syllabus. This report includes the final requirements document.

 3.2. Outcomes: What are the deliverables? How do we know when it is done?

There are three deliverables: preparation, demonstration, and report.

B) Hardware Design

Modified circuit diagram (SCH file), included in ZIP file

C) Software Design

Include the low-level I/O drivers , included in ZIP file

D) Measurement Data

Give any performance data you collected (none)

E) Analysis and Discussion (none)