Lab 5: Music Player and Audio Amp

Allen Pan & Paris Kaman

Requirement 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 music player. Educationally, students are learning how to interface a DAC, how to design a speaker amplifier, how to store digital music in ROM, and how to perform DAC output in the background. Your goal is to play your favorite song.

1.2. Process: How will the project be developed?

The project will be developed using the TM4C123 board. There will be two or three switches that the operator will use to control the music player. The system will be built on a solderless breadboard and run on the usual USB power. The system uses the 2 on board switches and 1 off-board switch. A hardware/software interface will be designed that allows software to control the player. There will be at least three hardware/software modules: switch input, DAC output, and the music player. The process will be to design and test each module independently from the other modules. After each module is tested, the system will be built and tested.

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

EE445L students are the engineers and the TA is the client. Students are expected to make minor modifications to this document in order to clarify exactly what they plan to build. 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.

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

The system will use the TM4C123 board, a solderless breadboard, and the speaker as shown in Figure 5.1. It will be powered using the USB cable. You may use a +5V power from the lab bench, but please do not power the TPA731 or the speaker with a voltage above +5V.

1.5. Terminology: Define terms used in the document.

SSI: Synchronous Serial Interface; A device to transmit data with synchronous serial communication protocol.

Linearity: linear relationship between two variables

Frequency response: magnitude (dB) vs. frequency (Hz); describes how a device responds to wave across a range of frequencies.
Loudness: the amplitude of the wave
Pitch: the frequency of the sound/wave
Instrument: the shape of the wave
Tempo: the number of beats per minute
Envelope: a function used to trim the shape of the note in order to make it sound more enjoyable and exhibit less clipping
Melody: refers to the main sequence of notes that is the focal point of a song
Harmony: refers to a second pitch that is a different wavelength from the melody. The note sounds more 'harmonious' if the amount of cycles before the sine waves reset is lower. Ie: Octaves sound the most harmonious because a note that is one octave up will fit 2 full wavelengths within one wavelength of the melody

1.6. Security: How will intellectual property be managed?

The system may include software from StellarisWare and from the book. No software written for this project may be transmitted, viewed, or communicated with any other EE445L student past, present, or future (other than the lab partner of course). It is the responsibility of the team to keep its EE445L lab solutions secure.

2. Function Description

2.1. Functionality: What will the system do precisely?

If the operator presses the play/pause button the music will play or pause. If the operator presses the play/pause button once the music should pause. Hitting the play/pause again causes music to continue. The play/pause button does not restart from the beginning, rather it continues from the position it was paused. If the rewind button is pressed, the music starts from the beginning. There is a mode switch that allows the operator to change the instrument being played for the music. There is a C data structure to hold the music. There is a music driver that plays songs. The length of the song should be at least 30 seconds and comprise of at least 8 different frequencies. 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 LCD output, this output should occur in the foreground. The maximum time to execute one instance of the ISR is 1.92 microseconds. You will need public functions Rewind, Play and Stop, which perform operations like a cassette tape player. The Play function has an input parameter that defines the song to play. A background thread implemented with output compare will fetch data out of your music structure and send them to the DAC. There must be a C data structure to store the sound waveform or instrument. You are free to design your own format, as long as it uses a formal data structure (i.e., struct). The generated music must sound beautiful utilizing the SNR of the DAC. Although you only have to implement one instrument, it

should be easy to change instruments. The song we chose to play is Sounds of Silence by Simon and Garfunkel

2.2. Scope: List the phases and what will be delivered in each phase.
Phase 1 is the preparation; phase 2 is the demonstration; and phase 3 is the lab report. Details can be found in the lab manual.

2.3. Prototypes: How will intermediate progress be demonstrated?
A prototype system running on the TM4C123 board and solderless breadboard will be demonstrated. Progress will be judged by the preparation, demonstration and lab report.

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 and obvious translation from sheet music to the data structure. 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 (note to students: you may edit this sentence to define a different style format). 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 three switch inputs. The DAC will be interfaced to a 32-ohm speaker.

2.6. Safety: Explain any safety requirements and how they will be measured.
If you are using headphones, please verify the sound it not too loud before placing the phones next to your ears.

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. The system consists of three switches used for pausing, resuming, and restarting the sound as well as changing the instrument of the notes. Two timers are used, one for the melody and one for the harmony. The reload values are calculated based on the frequency of the desired note.

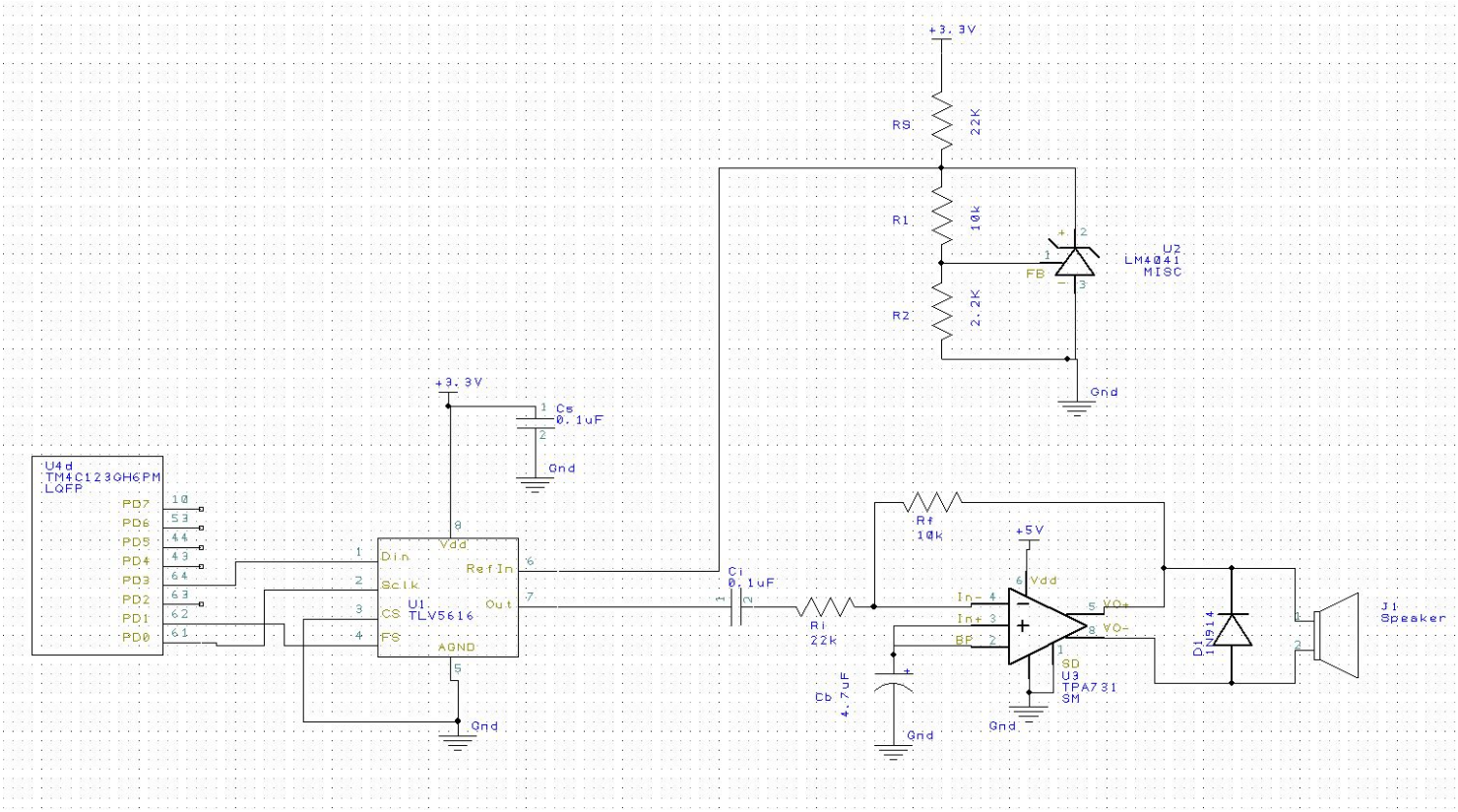
3.2. Audits: How will the clients evaluate progress?
The preparation is due at the beginning of the lab period on the date listed in the syllabus. The preparation consisted of demonstrating basic functionality of the modules

required for this lab, including making the notes in general, and the functionality of the switches. These were evaluated before moving forward with putting the modules together to make the full system.

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

The deliverables include this document along with our code. The preparation has been submitted on canvas and evaluated during the lab time on Thursday. The finished product of the lab was demonstrated during the lab session on Tuesday. The version of the lab that was tested on Tuesday is the version used for all measurements and submissions to canvas excluding the lab prep.

Hardware Design



Measurement

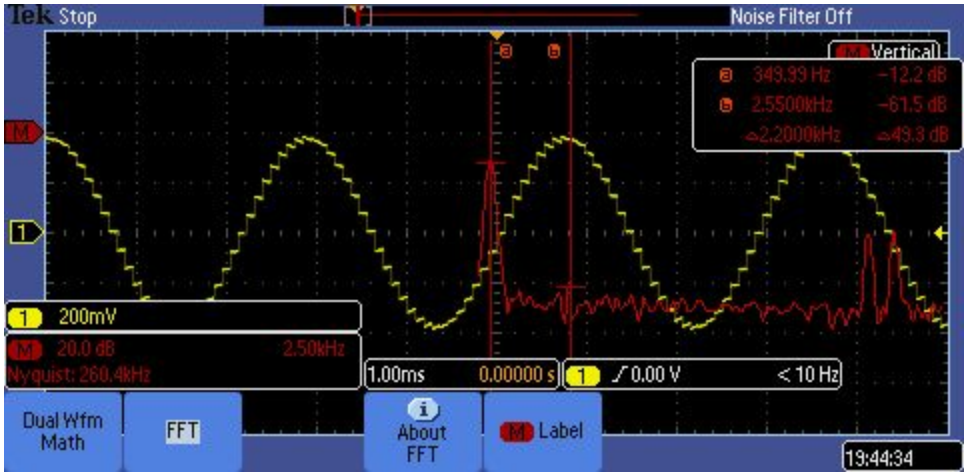
1) Show the data and calculated resolution, range, precision and accuracy (procedure 3)

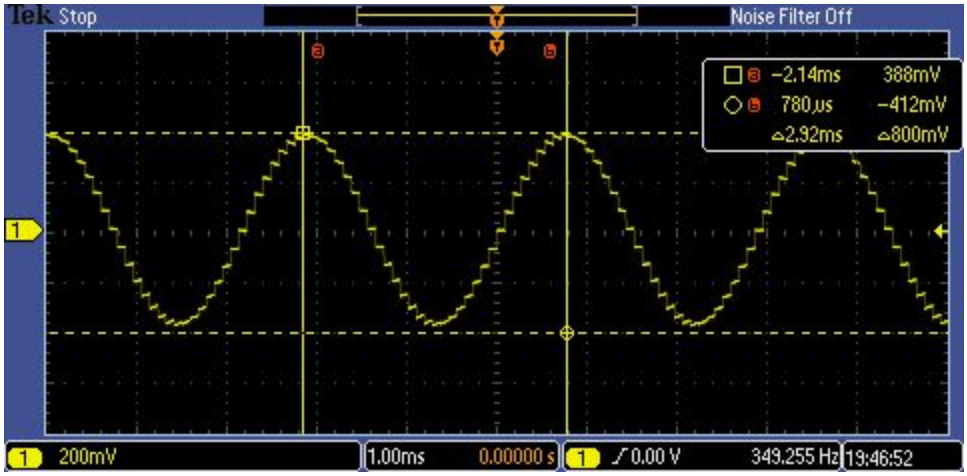
Our reference input voltage is 963 mV, so according to the datasheet, the voltage output from DAC is ranged from 0 to twice of 963 mV, i.e. 1926 mV.
We used a table size of 64, so our DAC takes 11-bit input.

SSI Transmitter Out	Measured Voltage (mV)	Expected Voltage (mV)
0	2	0
300	285	282
600	571	565
1000	952	941
1500	1418	1411
1800	1698	1694
2000	1892	1881
2047	1934	1926

Range: 2 - 1934 mV
Precision: 2048 alternatives, 11 bits
Resolution: (1934-2)/2047 = 0.944 mV
Accuracy for DAC input 1500: 1- (1418 - 1411)/1411 = 99.5%

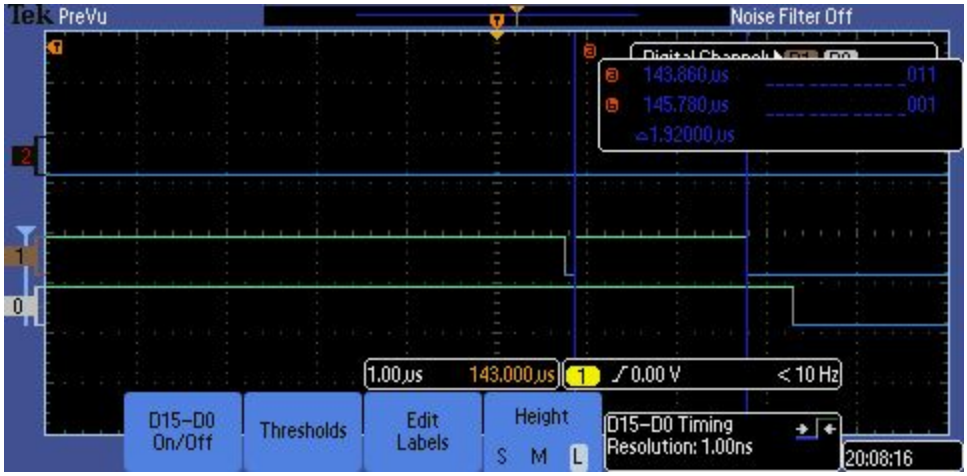
2) Show the experimental response of DAC (procedure 4) including SNR

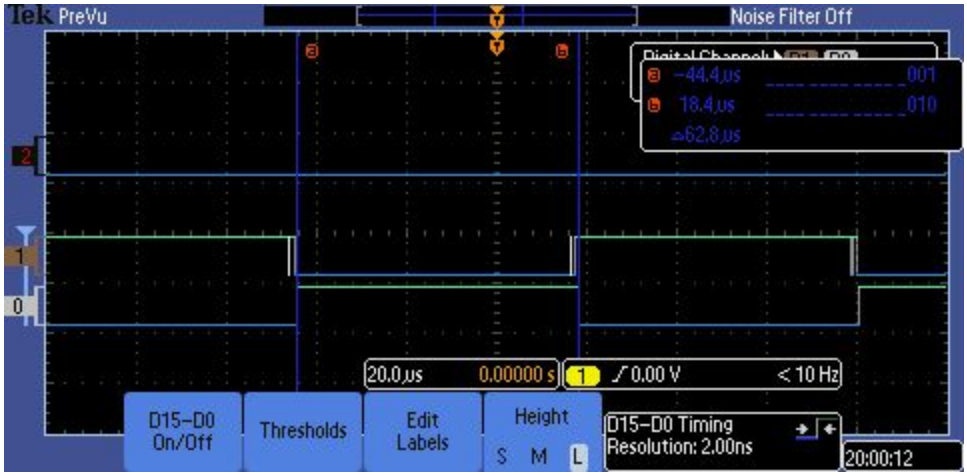




SNR = 49.3 dB.

3) Show the results of the debugging profile (procedure 5)

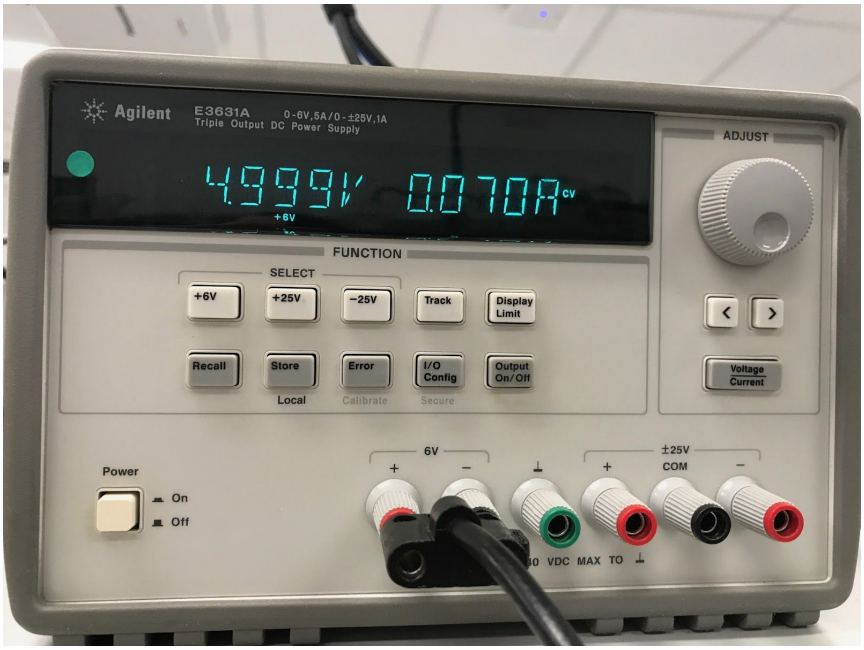




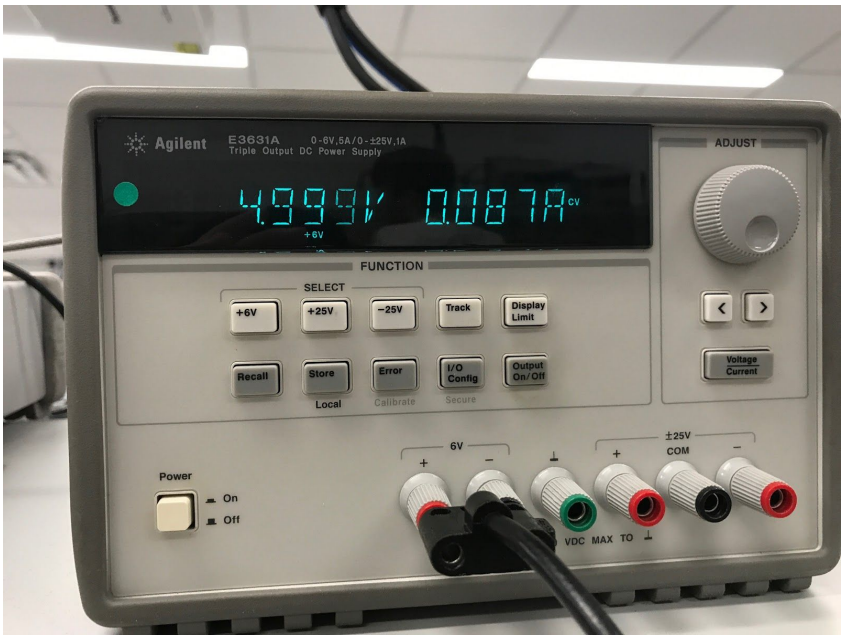
ratio of time spent in the ISR versus the main program is 1.92/62.8 = 3.06%

4) Show +5V voltage, voltageRMS (which will be very small) and current, with and without the music playing (procedure 7)

Without Music



With Music



RMS voltage is around 20 mv.

Analysis and Discussion

1) Briefly describe three errors in a DAC.

-nonlinearity- the resistance values are off in a way that makes the analog signal non-linear

-gain error- the DAC is linear but it's slope isn't the desireable slope to reach the full range of voltage with the full range of values

-offset error - the DAC starts at a non-zero voltage for an input of 0 from the digital end. This lowers the range of values you can represent

2) Calculate the data available and data required intervals in the SSI/DAC interface. Use these calculations to justify your choice of SSI frequency.

The setup time is 8ns, the hold time is 5ns, and the pulse duration is 25ns. So the time required for the data to be available is 38ns, that is around 26 MHz. The maximum serial clock allowed is 20 MHz, so we choose a value below it. In our case we used 10 MHz. The data will be available for longer than the time needed for the DAC input.

3) How is the frequency range of a spectrum analyzer determined?

Frequency range is from the minimum frequency of the input to half of the sampling frequency. Since any value above half of the sampling frequency isn't able to be sampled and would cause aliasing.

4) Why did we not simply drive the speaker directly from the DAC? I.e., what purpose is the TPA731?

TPA731 amplifies the current so that it is high enough to drive the speaker. The current coming directly from the GPIO pins is too low.