Daniel Diamont

Robert Noe

Lab 5 Report

1. 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 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 three switches that the operator will use to control the music player. One switch will play/pause the song; one switch will rewind the song; one switch will start the song initially. The system will be built on a solderless breadboard and run on the usual USB power. The system may use the on board switches or off-board switches. 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 sound controller. 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.

* Tasks:
  + Requirements document – Daniel Diamont
  + PCB Artist schematic – Robert Noe
  + Switch input software – Robert Noe
  + DAC Output software – Daniel Diamont
  + Sound controller software – Daniel Diamont
  + Checkout – Daniel Diamont and Robert Noe

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 – device to transmit data with synchronous serial communication. The clock is shared on both sides. Same as Serial Peripheral Interface (SPI).
* Linearity – the Digital to Analog converter must abide by the principle of linearity where its output is monotonic.
* Frequency response
* Loudness – Amplitude of the sound wave
* Pitch – 1/period of the sound wave
* Instrument – produces a particular sound wave according to its type
* Tempo – speed of the song
* Envelope – Amplitude vs. time. For notes to taper off, we can apply an exponentially decreasing envelope.
* Melody - a sequence of single notes that is musically satisfying.
* Harmony – the combination of simultaneously sounded musical notes to produce chords and chord progressions having a pleasing effect.
* Decibel (dB) – measure of the relative magnitude of two signals.
* Spectrum Analyzer – equipment that displays signal amplitude vs. frequency
* SNR - the ratio of the strength of an electrical or other signal carrying information to that of interference, generally expressed in decibels.

Definitions for the terms SSI, linearity, frequency response, loudness, pitch, instrument, tempo, envelope, melody and harmony can be found in the textbook. *(Note to students: add any addition terms you feel are needed)*

1.6. Security: How will intellectual property be managed?

The system may include software from 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 stops and the next play operation will start from the beginning.

There must be a C data structure to hold the music, which will be an array containing the notes of the song and the time that each note must be held for. There must be 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 xxxx *(note to students: replace the xxxx with performance measure of your solution).* 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.

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. *(note to students: you could use 8 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.

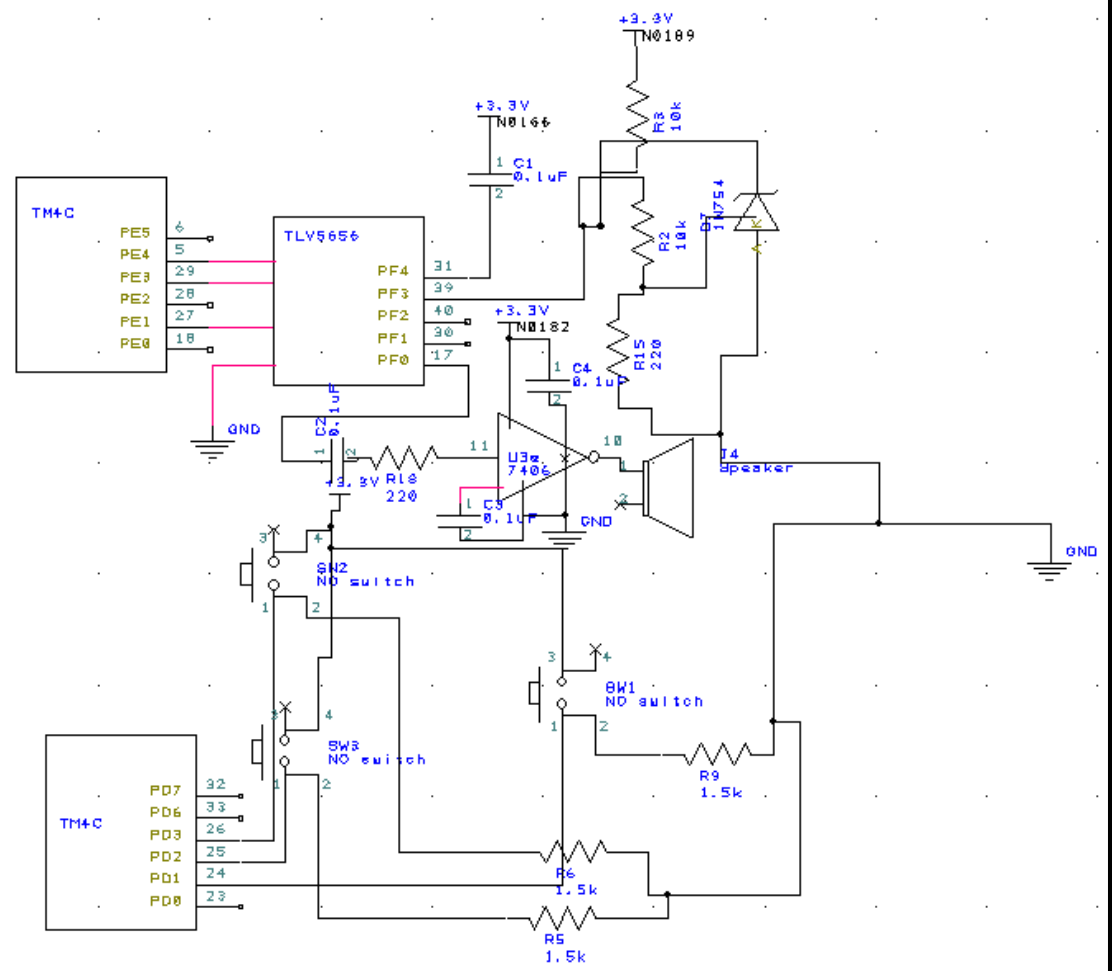
3.2. Audits: How will the clients evaluate progress?

The preparation is due at the beginning of the lab period on the date listed in the syllabus.

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

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

1. Hardware Design



1. Software Design
   1. Files committed on GitHub.
   2. Data flow and call graphs are the same as were proposed on the lab05 document.
2. Measurement Data
   1. Calculations



*Image 1: Resolution of our DAC output.*

**Resolution**:

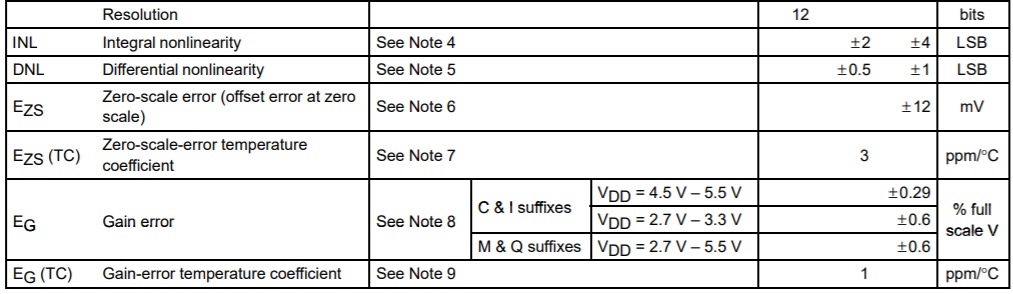
Our smallest visible step change on the oscilloscope appeared to be 40.0 mV.

Our calculated resolution is

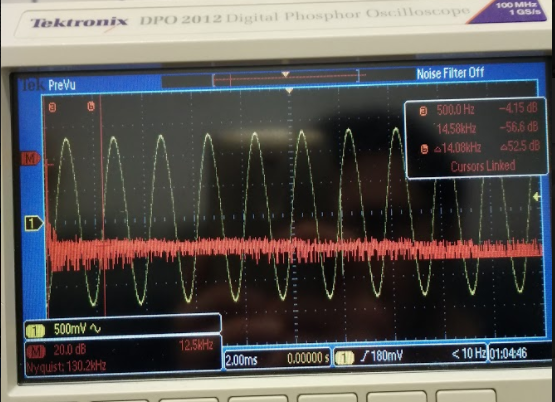
**Precision**: Number of alternatives = 4096 alternatives because the TLV5618A is a 12-bit DAC.

**Range**: Available DAC output range is 0V to 3.3V using a 3.3V power supply.

**Accuracy**: As shown on the DAC’s datasheet, the accuracy based on INL is typically

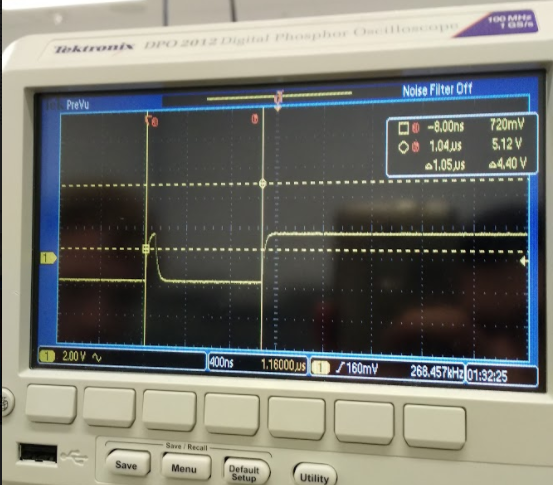


b) Experimental Response of DAC and Signal-to-Noise-Ratio:

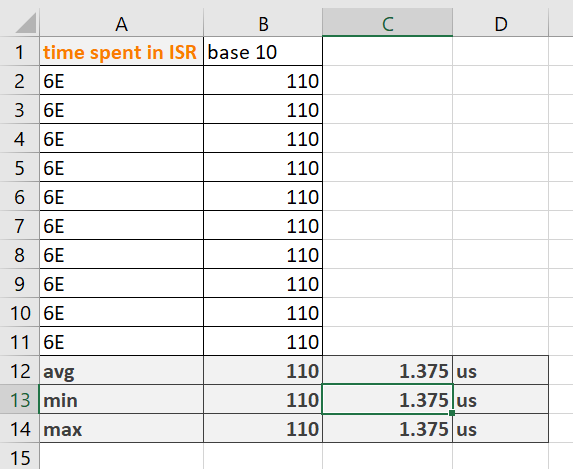


*Image 2: 500 Hz tone with SNR = 52.5 dB.*

c) Results of Debugging Profile – Time spent in ISR outputting to DAC

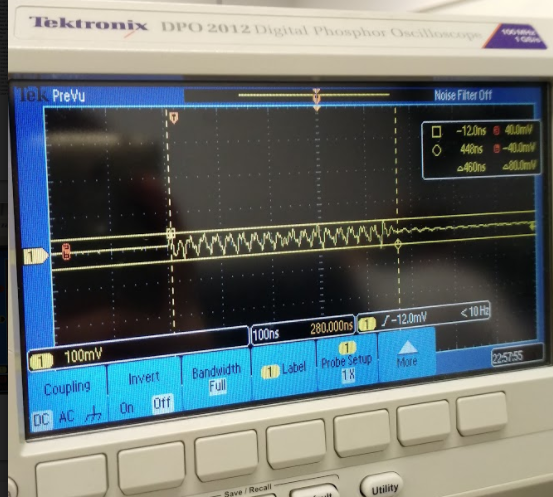


*Image 3: Time spent in ISR per oscilloscope measurement.*



*Image 4: Time spent in ISR per minimally intrusive software dump.*

d) +5V Power Line Noise RMS and Current with and without music playing.



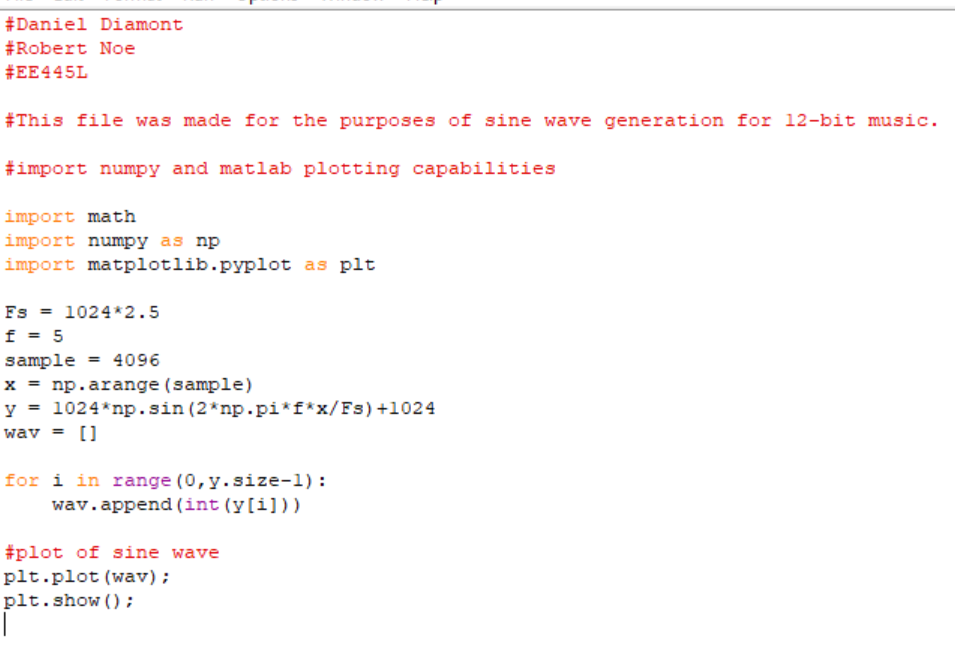
*Image 5: +5V Power Line Noise*



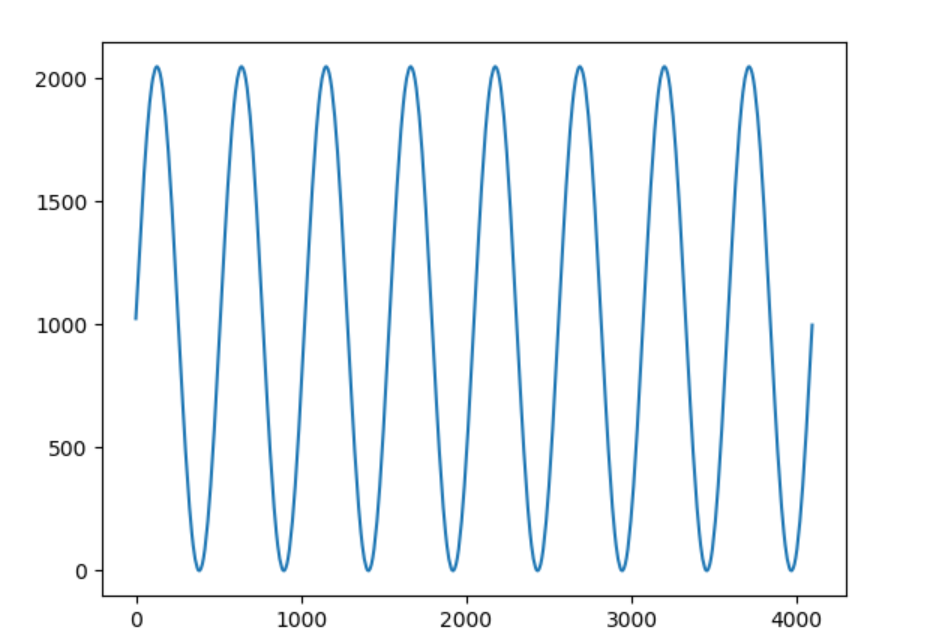
*Image 6: Current music OFF*



*Image 7: Current with Music On*



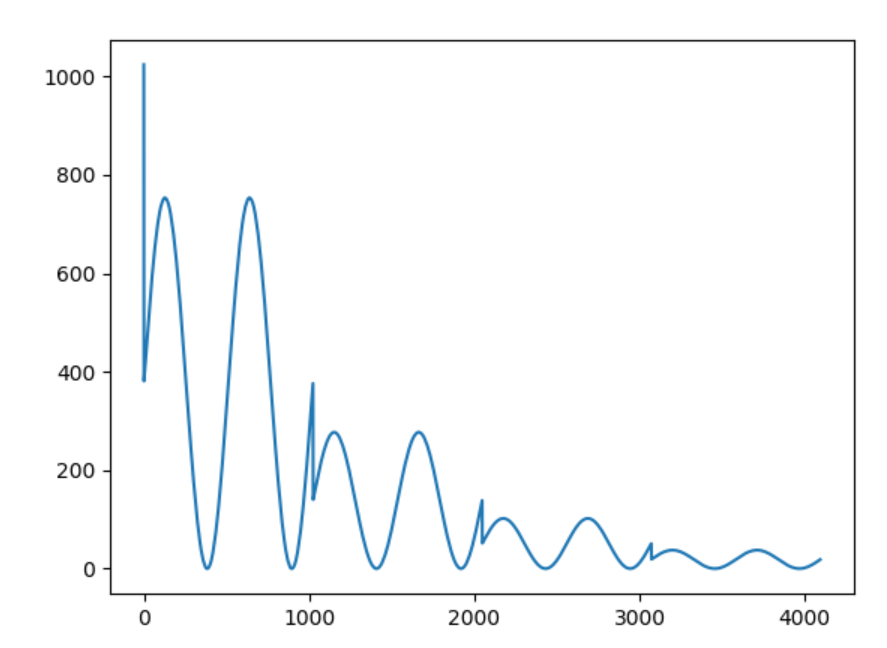
*Image 8: Code to generate wave-table with 4096 entries*



*Image 9: Wave-table with 4096 entries*



*Image 10: Adjustment to code to add exponentially decaying envelope*



*Image 11: Sinusoidal function with exponentially decaying function*

*Analysis and Discussion*

1. An aliasing error occurs when a digital signal is of a different frequency than the analog signal it’s meant to represent because it’s sampling the digital signal too quickly. A gain error is a shift in the slope of the output voltage versus the input. An offset error is a shift in the output voltage versus the input.
2. The data available interval must be greater than the time required interval and the set up time together. The set up and required times are both 10ns, so the data available interval must be at least 20 ns. Our clock half a period of 125 ns, which is more than enough time.
3. The frequency range of the spectrum analyzer is limited by the sampling rate.
4. The DAC doesn’t allow currents to travel through without damaging pins, so the TPA731 takes in a voltage value from the DAC so that the correct current travels through the speaker.