

A) OBJECTIVES

(used Requirements document of Lab 5)

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 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 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.

Definitions for the terms SSI, linearity, frequency response, loudness, pitch, instrument, tempo, envelope, melody and harmony can be found in the textbook.

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 stops and the next play operation will start from the beginning. There is a mode switch that allows the operator to control some aspect of the player. Possibilities include instrument, envelope or tempo. There must be a C data structure to hold the music. 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 `while(1){}` 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 200 ms. 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. Again, feel free to change the functional description to match your design. 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. 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 an 8-ohm or 32-ohm speaker.

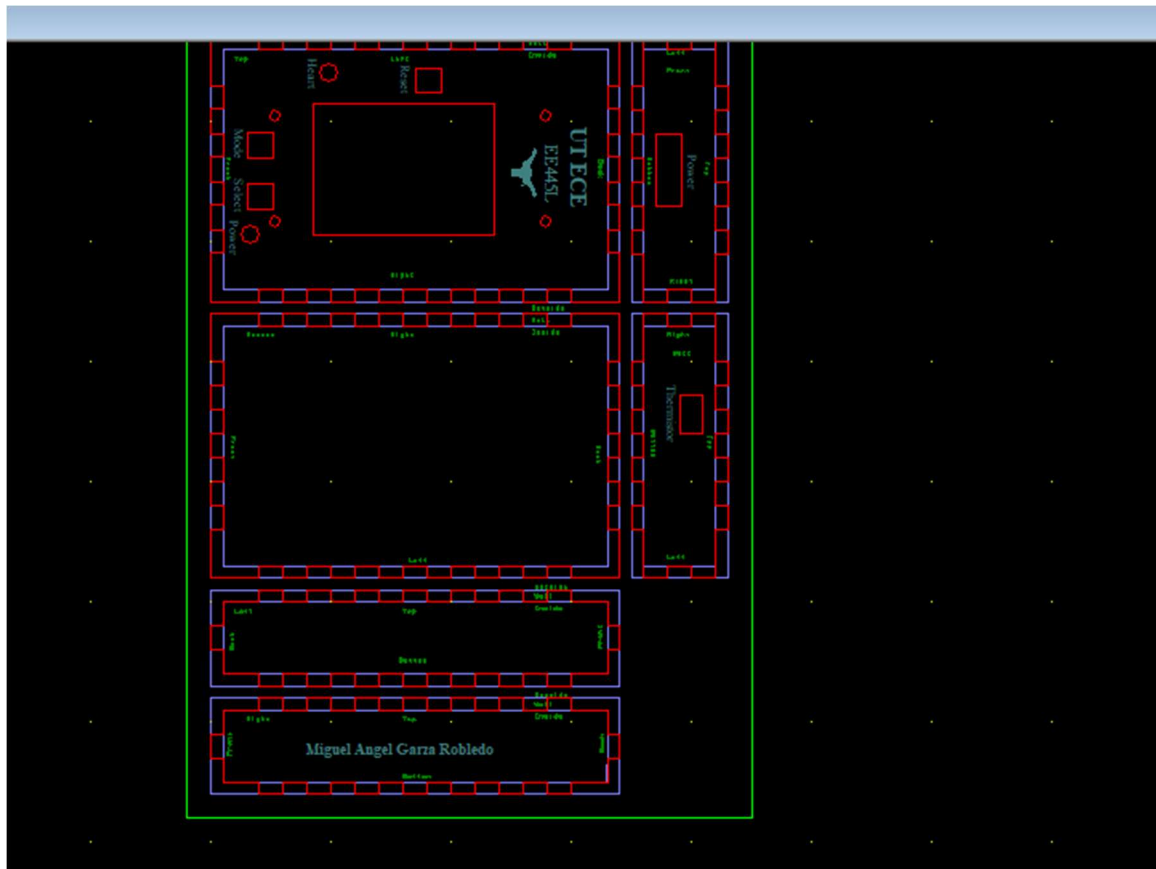
2.6. Safety: Explain any safety requirements and how they will be measured.

If you are using headphones, please verify the sound it not too loud before placing the phones next to your ears.

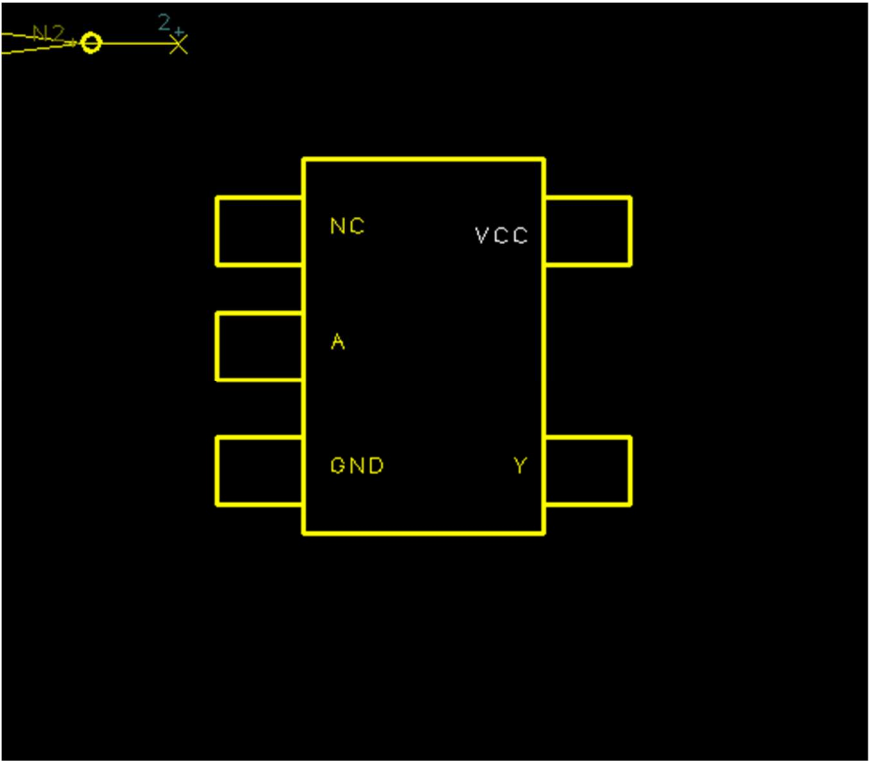
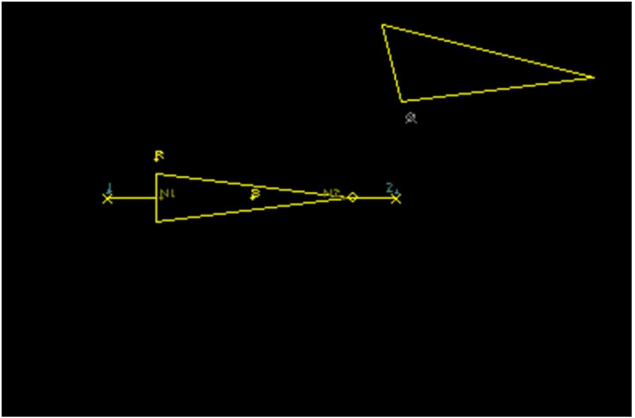
B) HARDWARE DESIGN

Description:

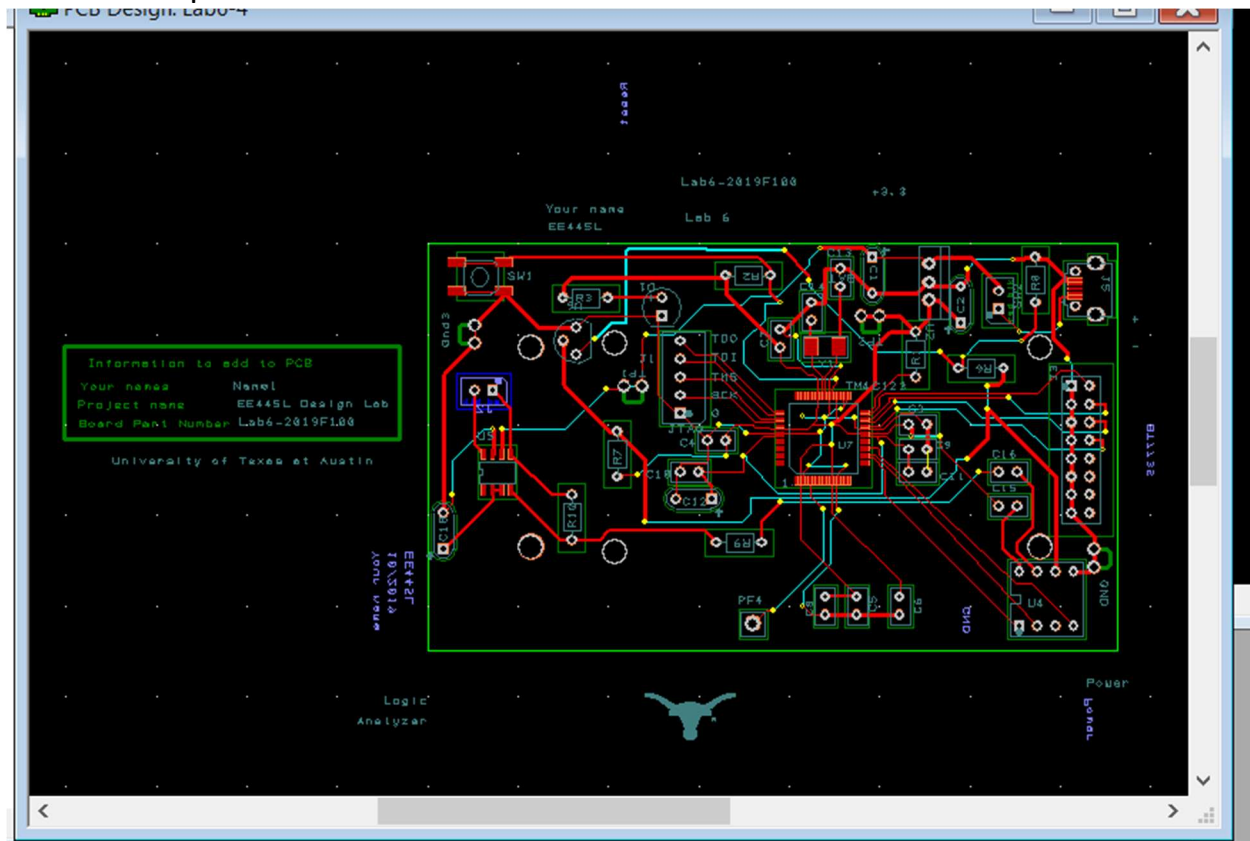
WoodBox.pcb is a 4.00 by 2.5 by 0.75 inch box, 125 mil wood, 6.5 by 9.5 inch raw material



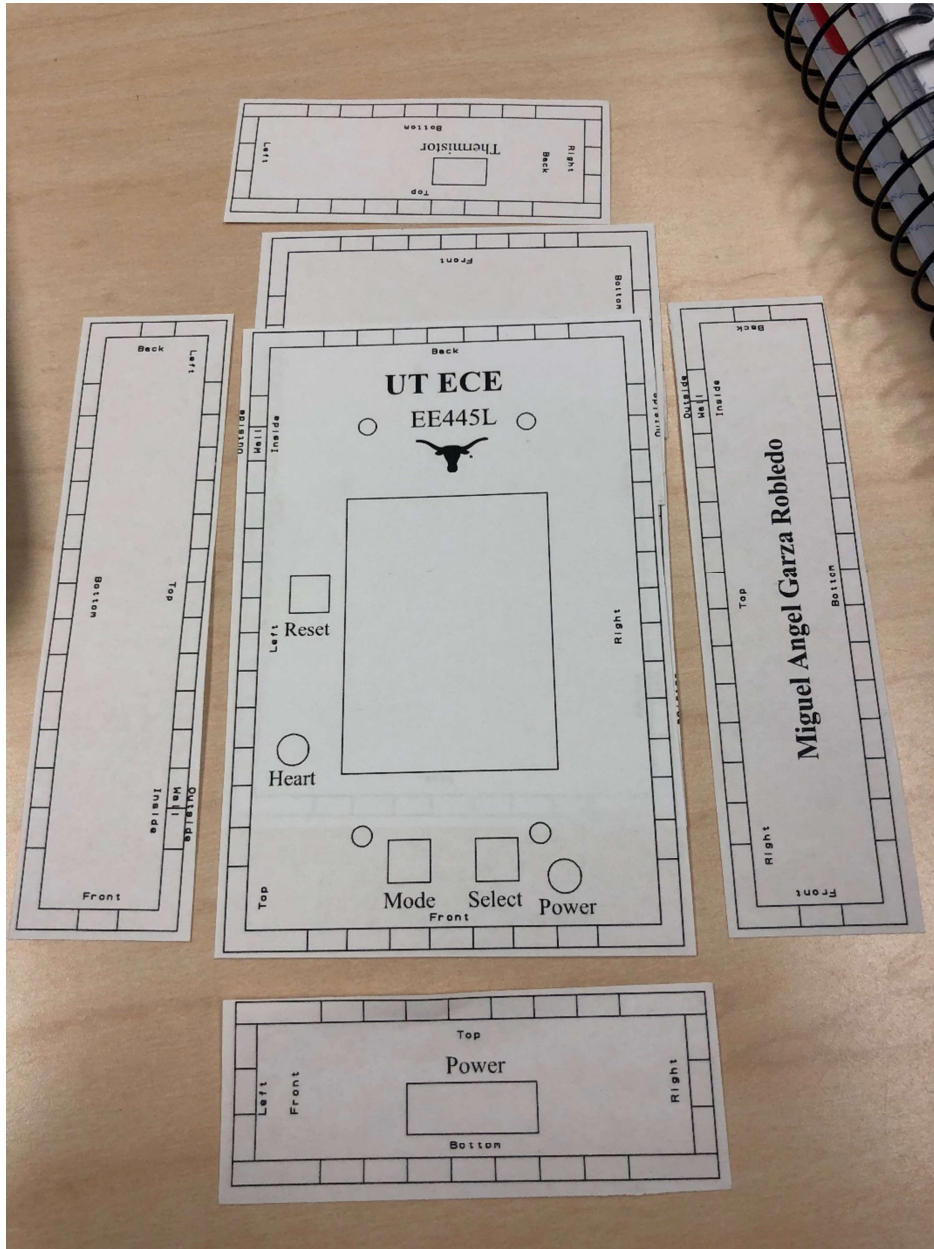
New component: (NOT GATE because previous IC I created got deleted 😞)



Final circuit picture:



Cardboard (paper) mockup:



Analysis and Discussion (pulled from Lab 5):

2.0 ANALYSIS AND DISCUSSION

2.1 Three DAC Errors

1. Remove access to a global variable
2. Implement mutual exclusion and allow only one thread to read from a global variable at a time. Possibly by disabling interrupts while executing the critical section

2.2 SSI Frequency

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

Interrupt maximum time measured by debugging LED (200 ms)

We choose a frequency that was set by systick timer, the longest duration

that it rested for was 200ms. Using this as our basis, any frequency that we

outputted to was at this frequency or much faster.

Figure 4: LCD
latency

2.3 Frequency Range of Spectrum Analyzer

How is the frequency range of a spectrum analyzer determined?

A **spectrum analyzer** measures the magnitude of an input signal versus frequency within the full frequency range of the instrument. The range is configured by sweeping the voltage-controlled oscillator through a range of frequencies of a by a band-pass filter, enabling the consideration of the full frequency range of the instrument.

2.4 Speaker Driver

Why did we not simply drive the speaker directly from the DAC? I.e., what

purpose is the
TPA731?

Theoretically you could, but the purpose of the TPA731 is to amplify the

sound

.