ECE5730 Final Project Proposal

Han Yang (hy592)
Tongyuan Liu (tl839)
Yibin Xu (yx623)

1. Project Description

This project aims to create an interactive music spectrum analyzer that leverages the capabilities of FFT algorithms to analyze musical pieces and visually present their spectrum on a VGA display. The RP2040 is responsible for audio playback (utilizing the knowledge we acquired in Lab1), communicating with the computer via USART to implement the user interface, implementing FFT algorithms, and communicating with the VGA display to show the spectrum.

1.1. User Interface and Functions

The user interface is structured across two main layers: music selection layer and playback control layer. Upon initiation, the user interface is set to music selection layer, presenting users with a catalog of musical tracks stored within the RP2040 storage. Users can browse through this list and make their selection by inputting the corresponding number of the desired music. Once a music is selected, the user interface transitions to the second layer, offering users the ability to play or pause the music, skip to the next one or revert to the previous one, and navigate back to the music selection layer. Meanwhile, the spectrum of the selected music is shown on the VGA display.

1.2. Fast Fourier Transform (FFT)

Fast Fourier Transform(FFT) is one of the most important algorithms ever developed. It is the basic algorithm in most digital communication, audio, and image compression. Compared with Discrete Fourier Transform(DFT)(O(n^2)), FFT(O(nlogn)) takes a huge advantage in speed and efficiency and produces the same functionality as DFT.

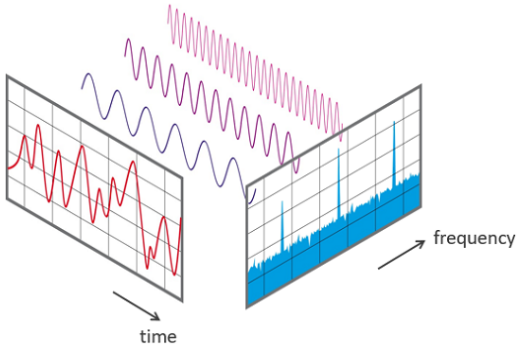

Figure 1 View of a Signal in the Time and Frequency Domain [1]

So what is the FFT? Suppose we have a time-series data of a signal, and we want to find out what frequencies are present in the signal. We can use the FFT to transform the data from the time domain into the frequency domain. Once in the frequency domain, we can analyze the amplitude and phase of different frequency components to understand the characteristics of the signal.

For instance, our dataset comprises three distinct signals: a DC offset with an amplitude of a 25Hz sinusoid of amplitude 0.8, a 50 Hz sinusoidal wave with a 0.7 amplitude, and a 120 Hz sinusoidal wave boasting an amplitude of 1.

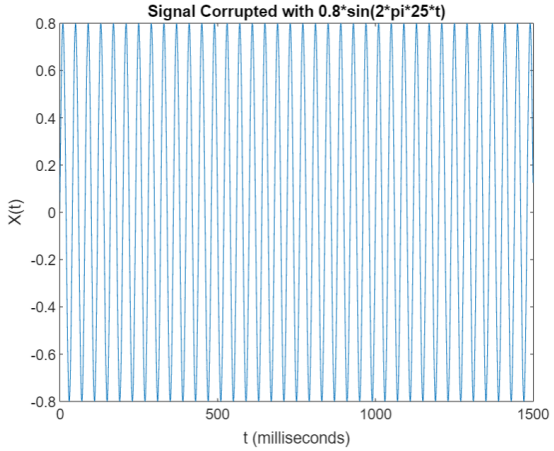

Figure 1 0.8*sin(2*pi*25*t)

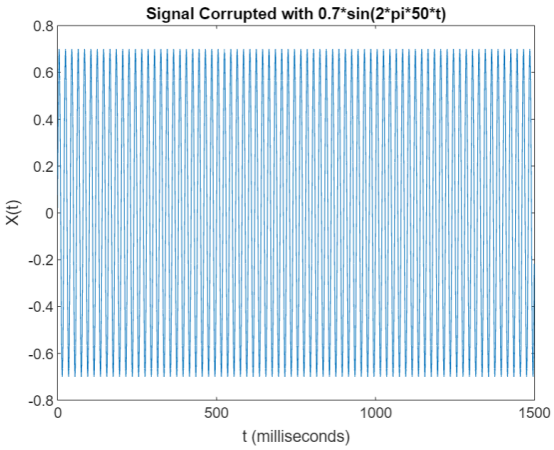

Figure 2 0.7*sin(2*pi*50*t)

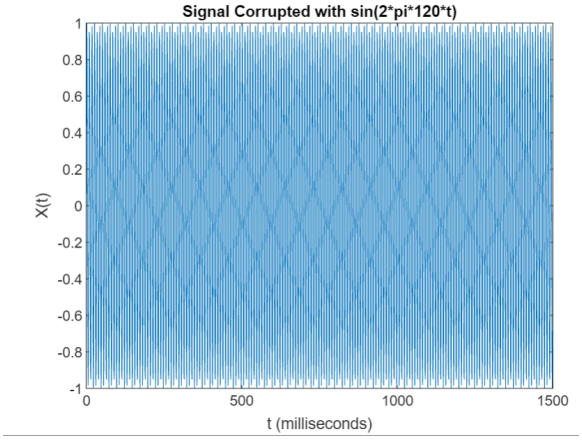

Figure 3 sin(2*pi*120*t)

S = 0.8*sin(2*pi*25*t) + 0.7*sin(2*pi*50*t) + sin(2*pi*120*t)

When we amalgamate the three individual signals, we generate a composite signal denoted as S in the time domain. But can we tell what is the original signal by just looking at the S signal? We can do FFT(S) for it to get a FFT spectrum. take the Fourier transform of the original signal and retrieve the exact amplitudes at 0.5, 0.7, and 1.0.

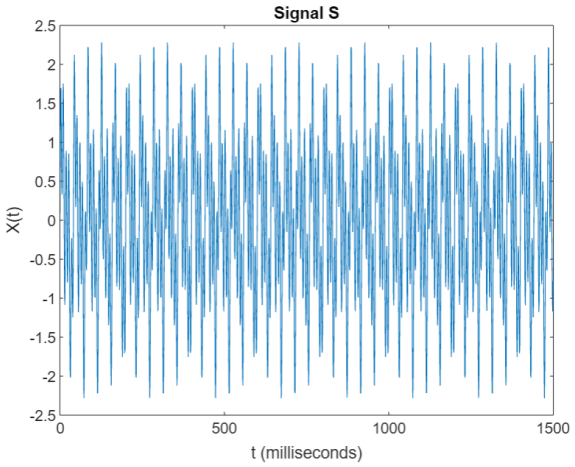

Figure 4 Signal S

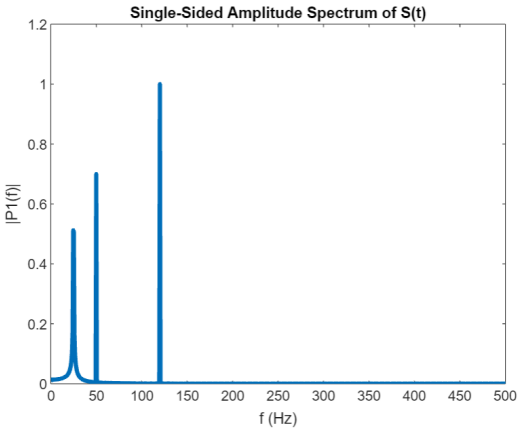

Figure 5 FFT Spectrum

How to implement FFT?

It is a complicated take to implement the FFT. but in simpler words, FFT is executed in two primary stages. In the first stage, the complete time series data is dissected into numerous fundamental two-point time series.

k	$n = 2^k$	permutation
0	1	0
1	2	0 1
2	4	0 2 1 3
3	8	0 4 2 6 1 5 3 7
4	16	0 8 4 12 2 10 6 14 1 9 5 13 3 11 7 15

Figure 6 Bit-Reversal Permutation [2]

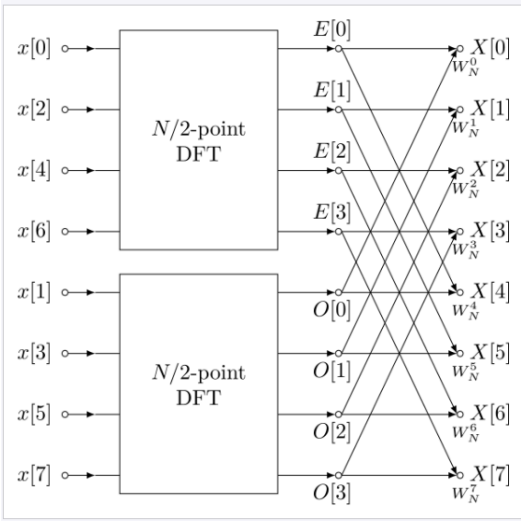

Figure 7 Butterfly Diagram [2]

Following this, the second stage involves converting these two-point time series into their corresponding frequency domain sequences. Subsequently, these two-point frequency domain sequences are meticulously amalgamated to construct the comprehensive frequency domain sequence.

2. Tentative Parts Cost List

A tentative parts cost list is shown in **Table 1.**

Item	Cost
RP2040	$0 (from lab)
Two speakers	$0 (from lab)
DAC	$0 (from lab)
VAG display	$0 (from lab)
Lab desktop	$0 (From lab)

Table 1 Tentative Parts Cost List

3. Social Impacts

This project brings music and technology together in a simple, user-friendly design. Using the straightforward USART communication protocol, our interface makes choosing and playing music easy for everyone, regardless of their technical ability. We will make sure to use only legally downloaded music to respect copyright rules. Our system's controls are simple to use, much like other common audio players, so they'll feel familiar to most people. We're also designing our visual displays to be clear and easy to see, with high-contrast animations that help those with limited sight enjoy the music, too. In short, our project offers a fun, accessible way for all to appreciate music through a blend of sound and visual technology.

4.1. Block Diagram of Music Spectrum Analyzer

The most important technology we need to implement is the Fourier Transfrom function. This function takes the real-time music signal as input, do the Fourier Transform operations as quickly as we can. The generated frequency data can be displayed in VGA to give a fancy visualization. The simplest block diagram can be as follows.

Figure 8 Simplified Data Flow Diagram

To have a real-time music spectrum, the music data input will be one sample at each time interval. However, the Fourier transform requires a sequence of data and will only generate the spectrum of this given sequence of data. Therefore, we need to have an array to store a

sequence of music data input. The data in this array is then fed to the FFT module to generate the spectrum.

The size of this array is important. To give a good visualization, we need to give a proper size of array. The reason is that the spectrum in visualization shows all information for a given time interval of the music input. A small array means a short time interval, the information displayed can be poor. A huge array can give a mass visualization result, as the information presented is too rich to be clear. At the same time, the execution time of FFT can be greatly influenced by the data size. The Fourier transform is of time complexity in O(n^2), Fast Fourier Transform (FFT) reduced the time complexity to O(nlogn). Complex number operations involving multiplication and division have an extremely high impact on performance. So we need to make a trade-off between execution time and the result of visual display.

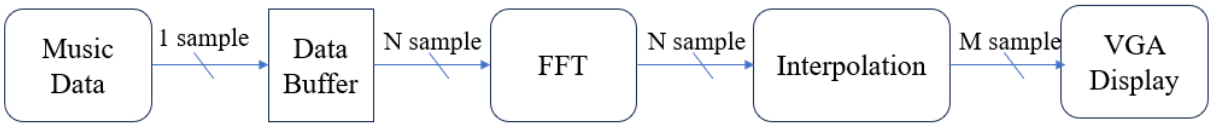

Figure 9 Detailed Data Flow Diagram

The number of acceptable samples of display may be about 128, or 256. Some computer vision cheats can also help us draw better spectral maps. For example, average value interpolation can easily double the samples in our spectrum display with almost no extra computation. So the final block diagram of our music spectrum analyzer can be as shown in Figure 9.

4.2. Tentative Source Code

There would be a rich open source library and git repository for FFT implementation in C and C++, which can provide us with a good starting point for our FFT implementation. The thing is, we are doing such a calculation-heavy program in a Raspberry Pi, we must apply some calculation improvement to reduce the execution time required for each display frame. We can exploit fixed-point data types with a good trade-off between accuracy and efficiency. We also can utilize parallel computation and overclock to provide more available computation time.

As for the VGA display, we can start with the demo code provided by Hunter for Lab 3. We need to modify the display frequency and display format to suit our visualization target. We may also want to add a more fancy display style that can be controlled by user interface.

4.3. Tentative Schematic For Hardware to Build.

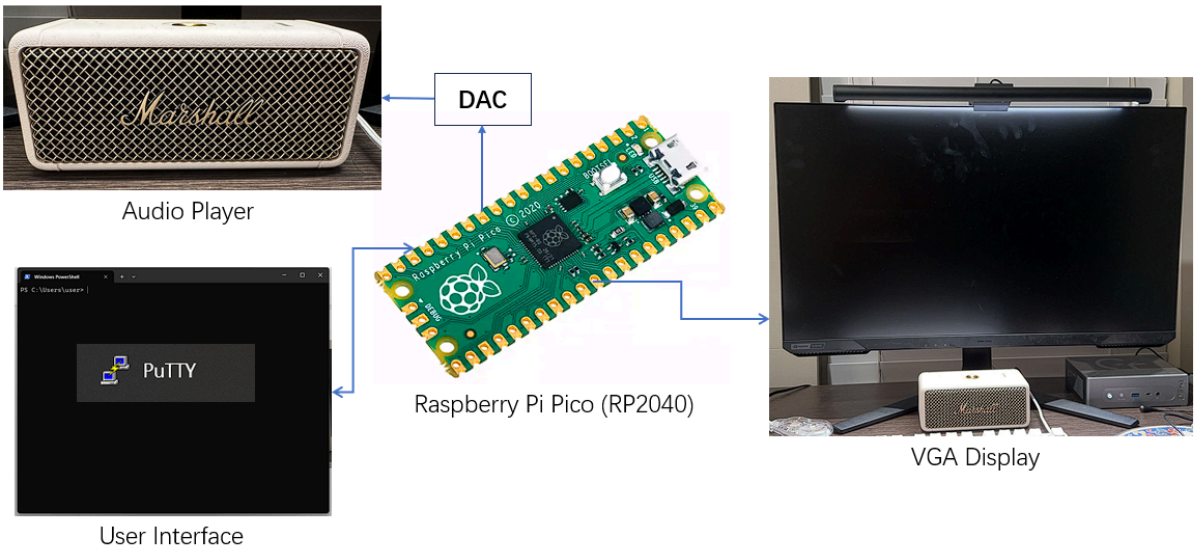

Figure 10 Hardware diagram

The hardware we need to build is not complex. We can combine the hardware connection for the audio player, the user interface and the VGA display from previous labs. The hardware schematic is shown above in Figure 10.

i) **VGA Display**

The VGA display is employed for our music spectrum visualization. It needs to support a color and real-time display. We may need to figure out a suitable size of data to store the color for each pixel.

ii) **Audio Player and DAC**

The digital music data is first sent to the DAC to convert into analog signal, then it will be played by the audio player.

iii) **User Interface**

The user interface is simply a terminal in a PC and connected with RP2040 with a serial connection.

4.4. Hardware Connections with RP2040

The following are the detailed hardware connections between RP2040 and other hardware.

HARDWARE CONNECTIONS

* - GPIO 16 ---> VGA Hsync
* - GPIO 17 ---> VGA Vsync
* - GPIO 18 ---> 330 ohm resistor ---> VGA Red
* - GPIO 19 ---> 330 ohm resistor ---> VGA Green
* - GPIO 20 ---> 330 ohm resistor ---> VGA Blue
* - RP2040 GND ---> VGA GND

SERIAL CONNECTIONS

* - GPIO 0 --> UART RX (white)
* - GPIO 1 --> UART TX (green)
* - RP2040 GND --> UART GND

DAC CONNECTIONS

* - GPIO 5 ---> CS_bar
* - GPIO 6 ---> SCK
* - GPIO 7 ---> SDI
* - GPIO 8 ---> LDAC

Reference

[1] NTI Audio. (n.d.). FFT Time Frequency View [Image]. NTI Audio. https://www.nti-audio.com/portals/0/pic/news/FFT-Time-Frequency-View-540.png

[2] Wikimedia Foundation. (2023, October 10). *Fast fourier transform*. Wikipedia. https://en.wikipedia.org/wiki/Fast_Fourier_transform

[3] Adams, V. H. (2023). *Demo Code for the Raspberry Pi Pico* [C].

https://github.com/vha3/Hunter-Adams-RP2040-Demos (Original work published 2022)