ECSEE 4840 Embedded System Design, 2019 Spring
Project Presentation

Musical Stimulus Visualization

Guanxuan Li (gl2619)
Hongyu Zou (hz2552)
Shanglin Guo (sg3640)
Yiqi Sun (ys3127)

Summary Description

Our project:

- Inspired by the patterns shown on the old generation MP3 screen that corresponds to the pitch and volume of the music being played.
- Achieve real-time processing as well as the video output generation.
- Reacts to a real-time sound input from USB microphone.
- The wave will spread out from the center and fade out. The color of the circle corresponds to the frequency and the initial size of the pattern corresponds to the volume.

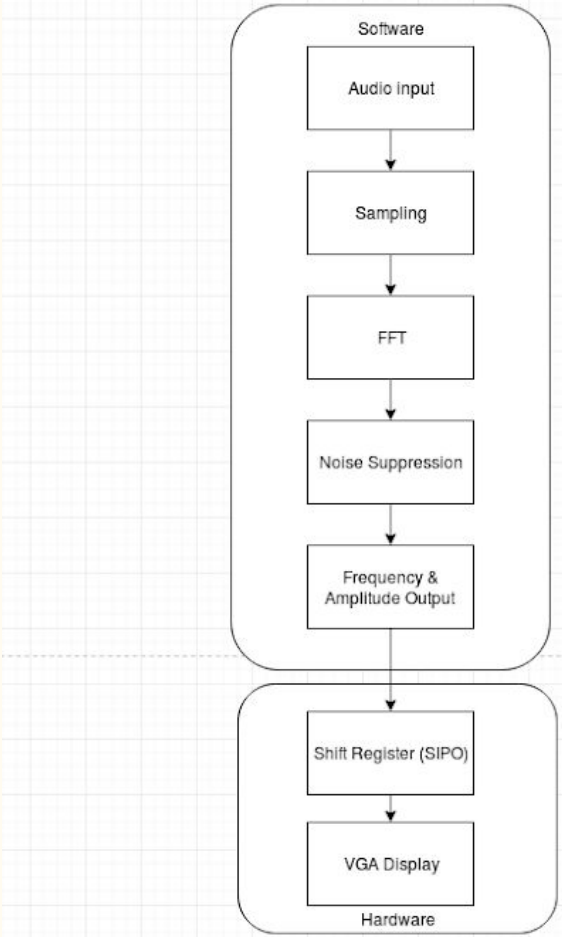
Challenge

1. Obtaining data from a USB microphone.
2. Raw data decoding and processing of the audio information.
3. Figuring out a way to implement good musical data visualization scheme on the monitor.

Flowchart

The audio input is collected from a USB microphone and handled by the C code running on the hard processor system (HPS). The audio input is sampled at a constant interval (~ 0.4s). After FFT and noise suppression, the resulting frequency and amplitude information is passed to the hardware through Avalon Bus.

The hardware component read the data from software and store it in SIPO shift registers, then the parallel output from shift registers is used for visualization, and the result is displayed on a 640x480 VGA monitor.



Input of Software

For data input, we have planned several options.

From shell command “arecord”, record to a wav file.

- Output the microphone data to a ‘.wav’ file and then process the file data with FFT. These two steps are implemented in 2 threads for recording and processing.

- **Real-time recording and FFT**

  Directly obtain the microphone data from a USB port using c program and do data processing on the input directly.

- Last resort: read pre-recorded “.wav” file to generate visualization output on the display and play the sound at the same time using SDL2.

Software

Audio Input

Recompile Kernel.

- The original kernel doesn’t have USB sound drivers to support audio input.
- A new kernel containing these drivers is implemented.

Install ALSA.

- A software framework that provides an application programming interface (API) for sound card device drivers.
- Alsa provides audio and MIDI functionality to the Linux operating system.

Sampling

“alsa/asoundlib” library has been used for real-time audio input reading.

- Sampling rate is set as 44100Hz.
- PCM_FORMAT is set as ‘S16_LE’ (signed 16 bits in little endian).
- Mono audio channel

Input data decoding.

- Each sample is a signed 16 bits in little endian having high 8bits and low 8 bits.
- Data is stored in the form of complement.
- Decode the data to provide correct time-domain signal for FFT.

FFT

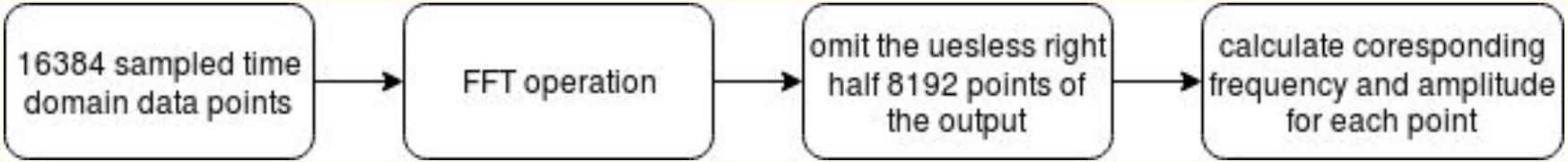
A fast Fourier transform (FFT) is an algorithm that computes the discrete Fourier transform (DFT) of a sequence, or its inverse (IDFT). Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa.

FFT is used to obtain the frequency information from real-time audio signal input.

“fftw3” library has been used for our FFT operation.

FFT

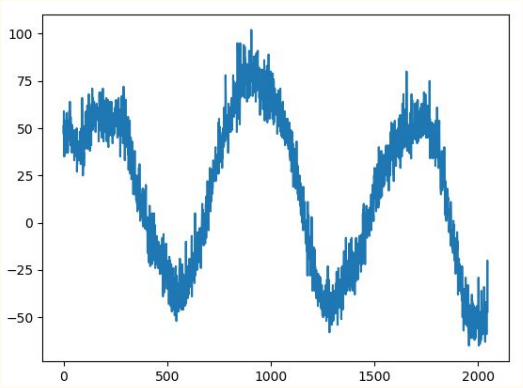
FFT flow chart



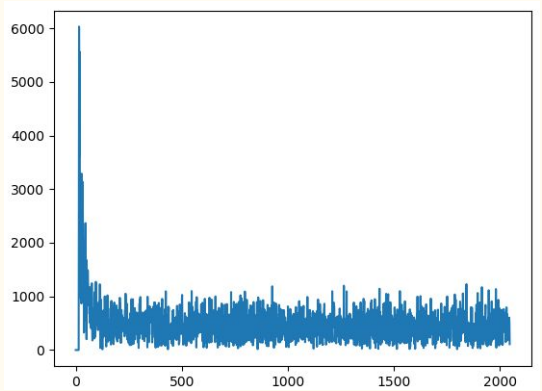
FFT

Test Results

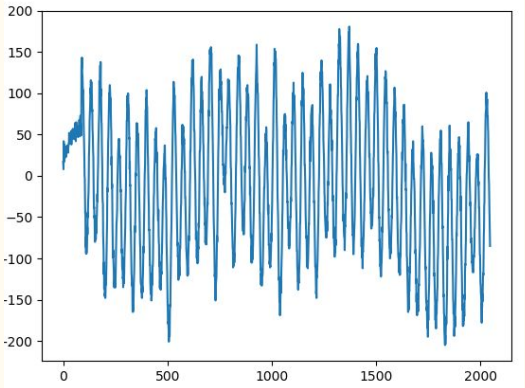
Noise



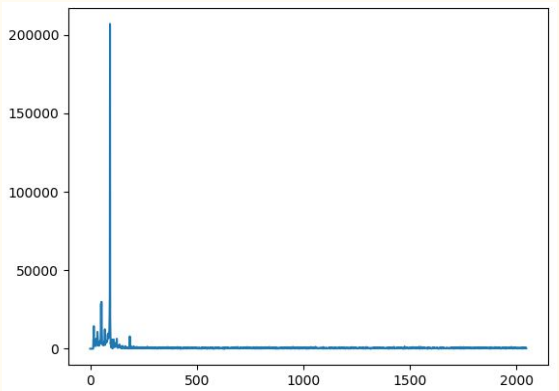
FFT



1K test sound



FFT



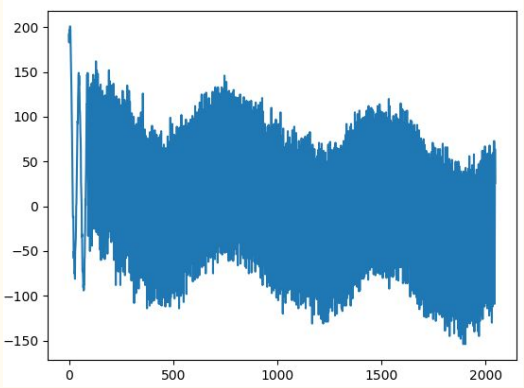
Time Domain

Frequency Domain

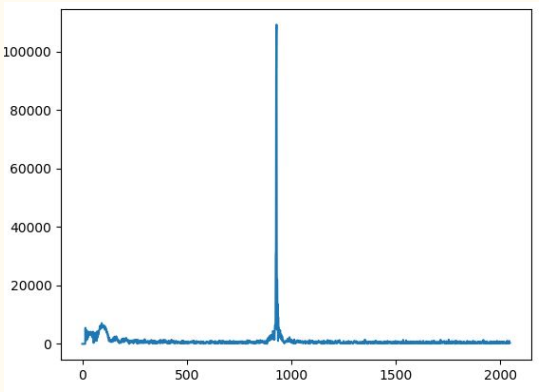
FFT

Test Results

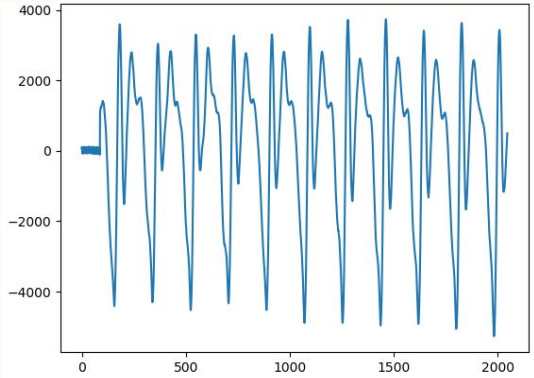
10K test sound



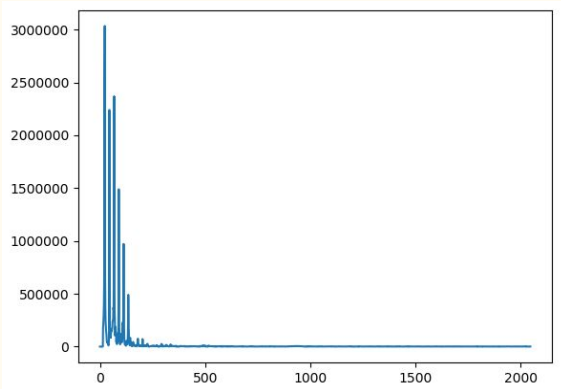
FFT



Human sound



FFT



Time Domain

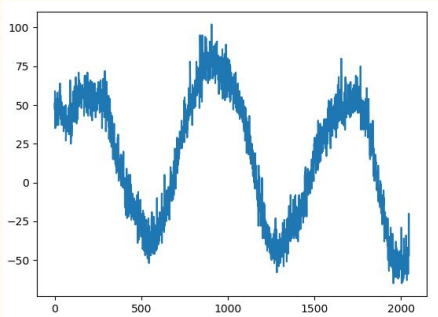
Frequency Domain

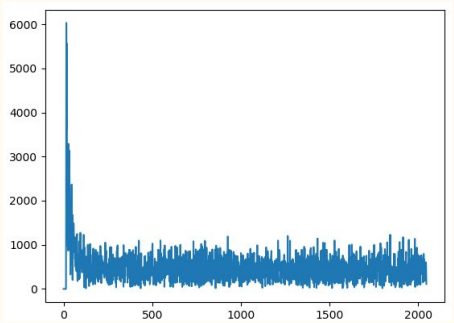
Noise Suppression

As environment noise will have negative impact on the FFT result of useful data, noise suppression is required.

- Sample the noise and operate FFT.
- Figure out the main distribution of noise energy in frequency domain.
- Spectral subtraction is used to omit the useless noise in frequency domain.

Noise





Frequency & Amplitude Selection

Since there are a large amount of FFT output points, it is unnecessary to transmit all of them to the hardware. So a selection algorithm should be implemented to select some valuable data.

The whole frequency range is divided into 4 segments. In each segment, choose one frequency having the largest energy. Then four <frequency, energy> pairs is transmitted to the hardware.

Moreover, as the raw frequency & energy values of the FTT output are too large, modulation should be done to adapt them to meet the hardware requirement. (Logarithm Compression)

Circular Movement

The 4 dynamic circles are moving around the center of the screen.

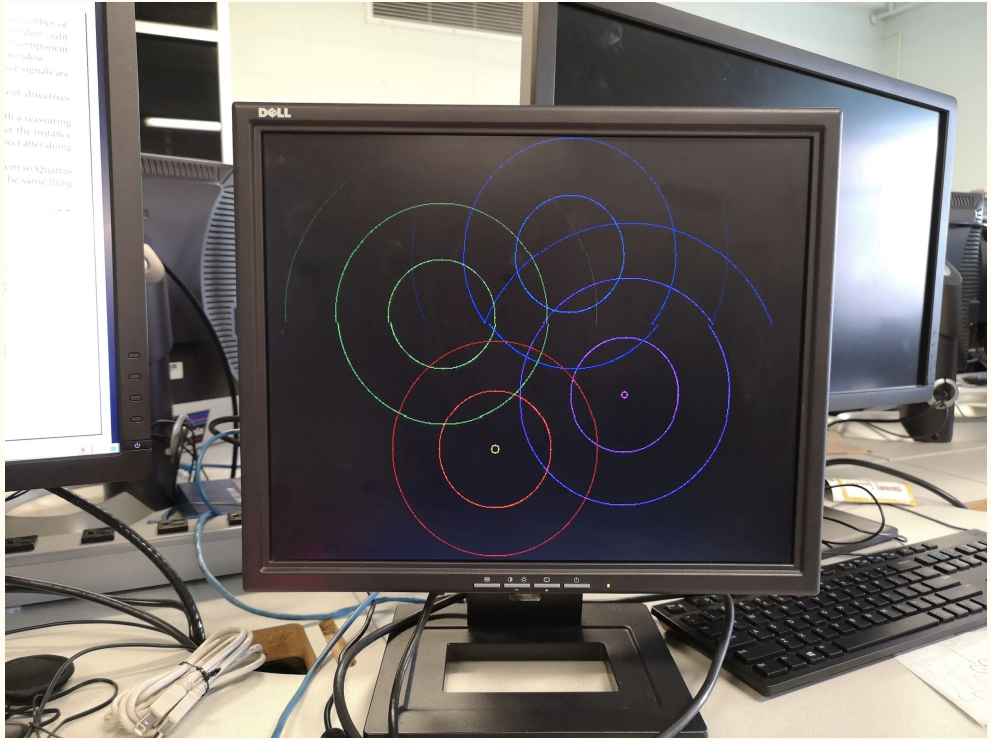
The circular movement algorithm is implemented in the C program.

The positions (coordinate) of the 4 dynamic circles are generated in real-time and passed to the hardware.

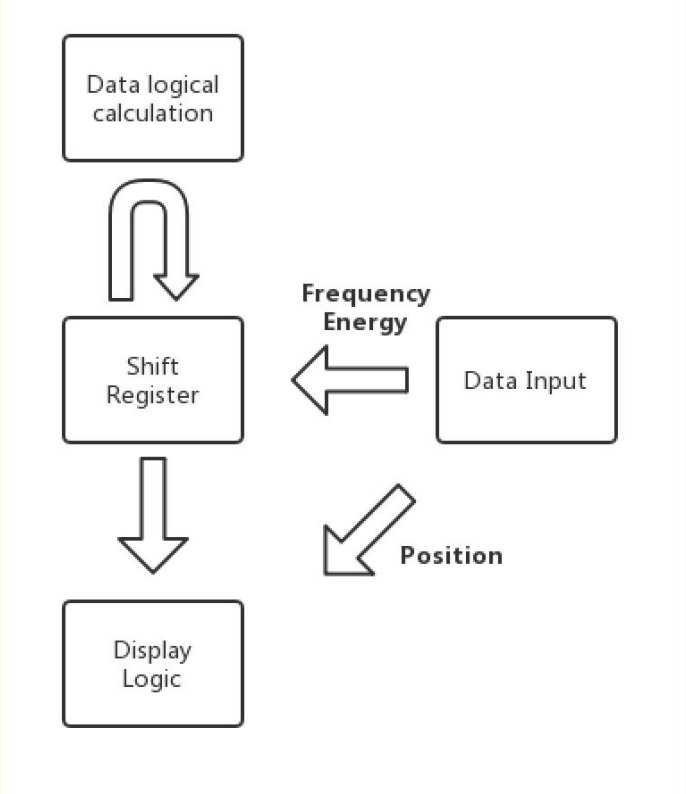
All the data are transfer to the hardware through ioctl().

Hardware

Basic Pattern



Design Logic



Display Pattern Logic

1. When enable signal is 1, new 16 digit frequency and size data are written into the right side(lower side) of shift registers.

2. Every time tictok signal goes in, every 16 digits of the shift register would increase(size) or decrease(frequency) once.

3. Every clk50 cycle, hcount and vcount would change and new position information is passed into the hardware. Thus, the circles would appear simultaneously.

Frequency & Amplitude

Frequency denotes color, color decrease to zero.

Input: Amplitude of signal of corresponding frequency intervel.

Cycle input of center points of circles.

Output: Circles which fade out gradually and color/size corresponding to input amplitude and frequency. Circles would also rotate.

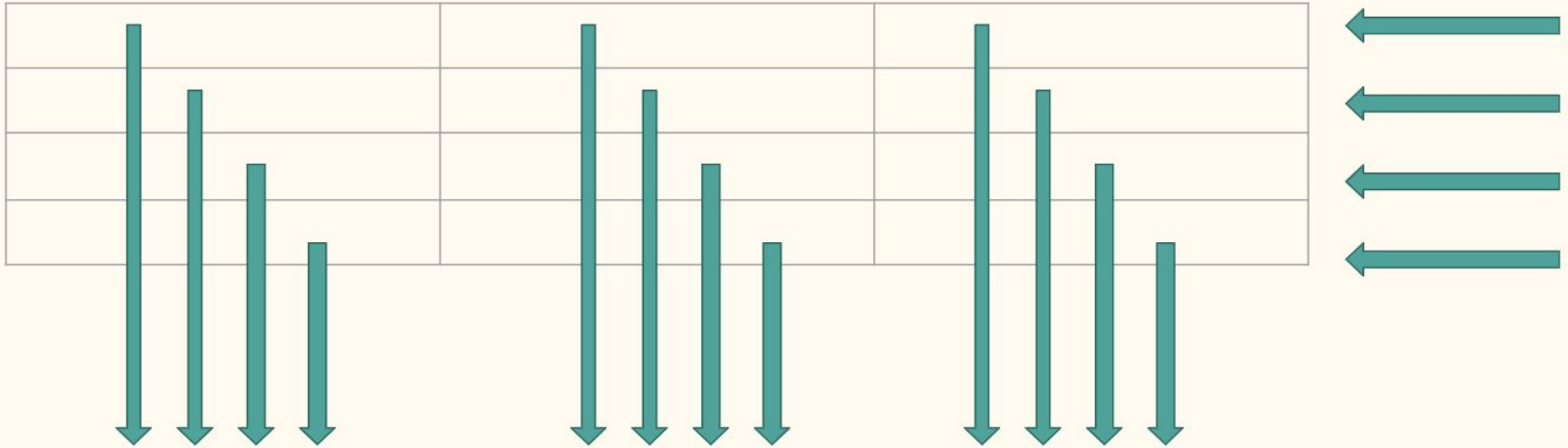
Shift Register

- 4x3 shift register.
- For storing historical data .
- Serial input, parallel output.
- Controlled by write enable signal and tiktok signal.
- Store the frequency and size of not yet faded circles as new data comes in.
- Tic tok signal is generated every 300000 clk50 cycles.

Shift Register

4x3 left shift register

4x16 bits data in



4x48 bits data out

Demo