FPGA Piano-Playing Robot

Brendan Ashworth, Max Hardy, Anthony Nardomarino

Table of Contents

Description

The purpose of the FPGA Piano Playing Robot is to create a dynamic machine capable of listening to a user play a tune, extract the important frequencies from the chords or melody, and to assemble and send instructions to a set of robotic hands capable of reproducing the tune on an anchored keyboard or piano. Living in three states, the robot will first be able to listen to an audio input and dissect the harmonic equivalent frequencies from prominent sounds; it will then be able to plan its route: determine when to play what chords at what location on the keyboard. Finally, it begins to play the song on the keyboard on a loop based on its perceived sounds and generated instructions for how it will perform.

Block Diagrams

Our model operates under three overarching system states: listening to an Audio Input by the Operator (Audio Analysis & Storage), organizing a generated musical output into a strategy for play (Routing & Strategy), and finally actuating a lattice of PLA fingers to play the analyzed music on a real anchored keyboard or piano.

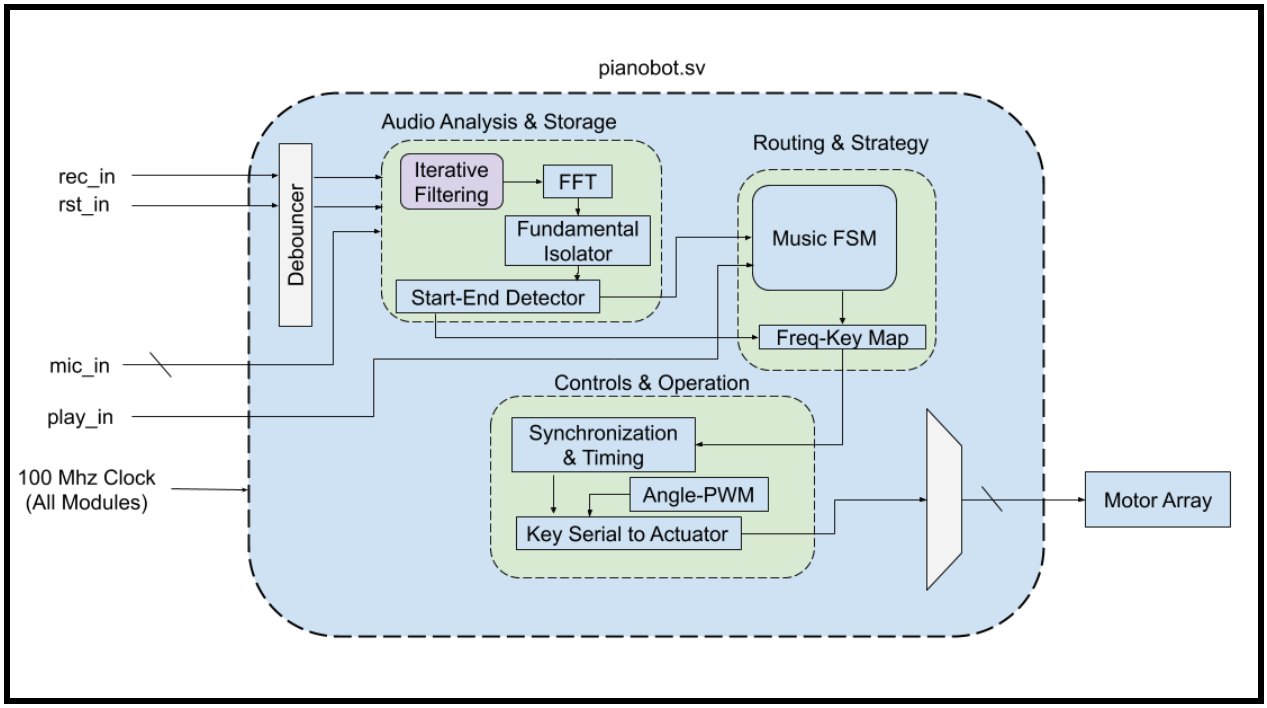


Figure 1: Modular System Architecture

System Architecture

Audio Analysis & Storage

Audio input from a microphone should be able to be analyzed not only from the keyboard, but from other instruments, and even prerecorded audio. With this in mind, an audio sample is taken by the Audio Analysis & Storage module, and is processed using Fourier Analysis to create a map of which frequencies are prominent at any given time. Two BRAM blocks are used to store audio for processing, and another to store frequency information after an FFT, volume filtering, and piecing together of output.

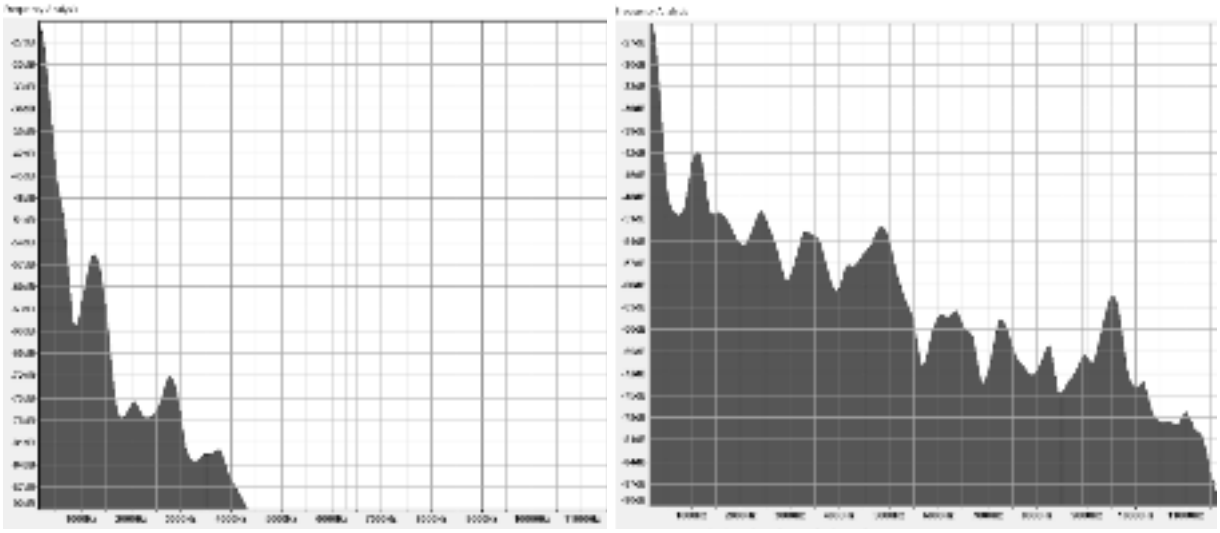
FFT: Fourier Decomposition

The purpose of our Fourier Decomposition is to analyze a segment of an incoming sound wave coming from a microphone input, and to break it down into the sine waves that compose it using Fourier Analysis. By analyzing the amplitudes and frequencies of these sine waves, the system will then be able to deduce harmonic patterns and the prominence of each frequency, assisting in the process of generating musical notes from a sound wave.

The stored audio input is split up into several pieces to more accurately analyze individual note and chord frequency information using a Fast Fourier Transform. The result will be a two-dimensional array-like structure containing a combination of frequency and time information, or a "spectrogram".

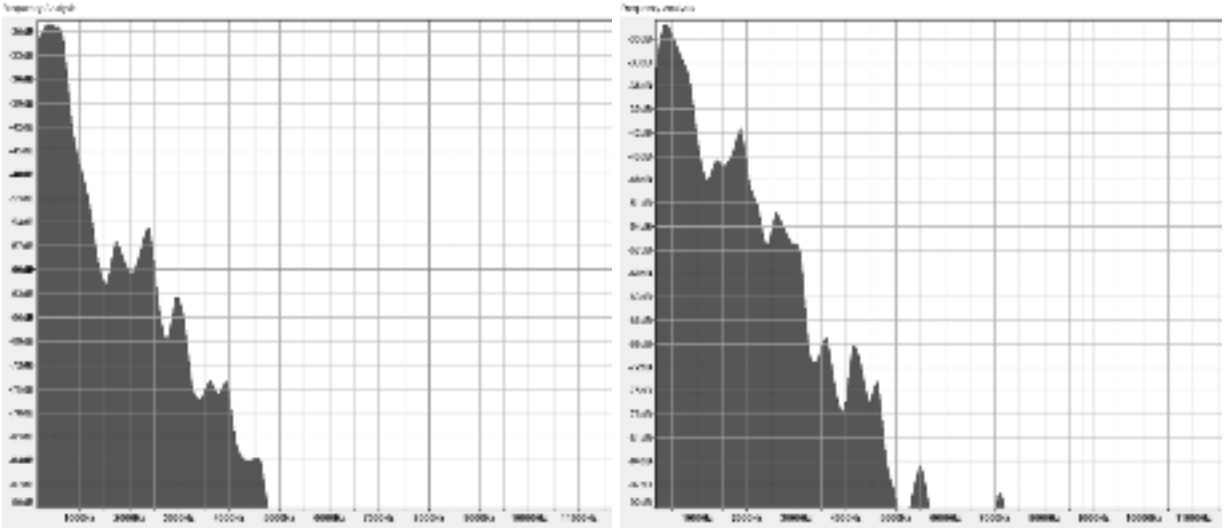
Below can be seen the recorded fourier transform differences in a Guitar's harmonic information vs. a Piano's harmonic information based on singular note play and a chord. Note to chord relationship is described by a DN major chord consisting of DN, F#N, and AN.

Guitar Fourier Transformations[1]
(D4 note vs D4 Major chord)

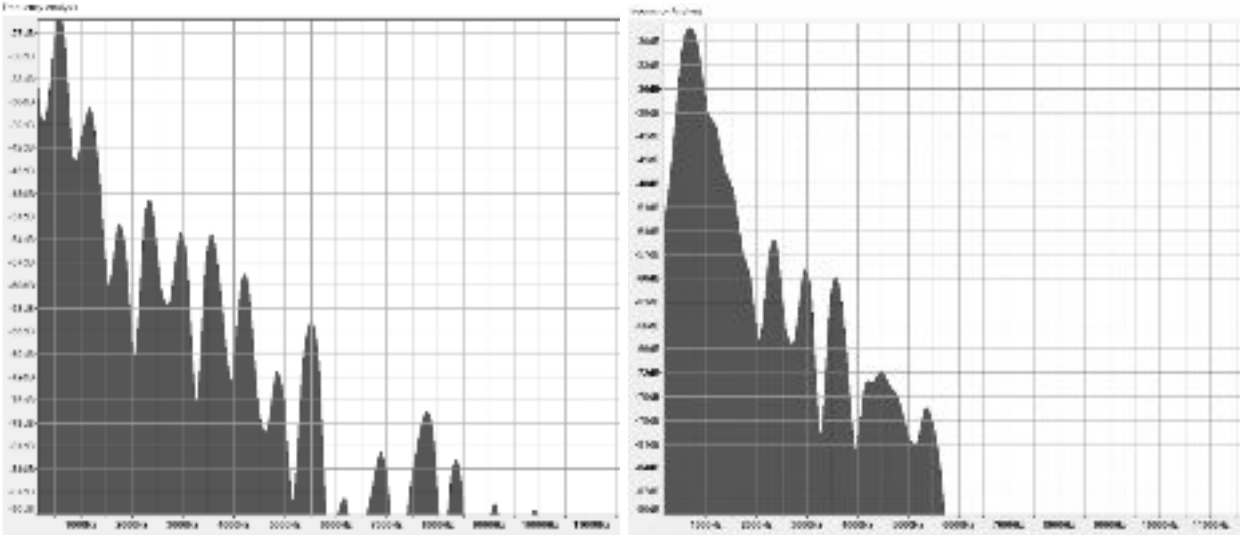


Piano Fourier Transformations
(D4 note vs D4 Major chord)

[1] Transformations generated by Audacity's Frequency Analysis interface.



D5 Note vs. D5 Major Chord



Fundamental Isolator

This module will acquire the fundamental frequencies. Overtones (harmonic information), are not relevant to the notes that must be played to replicate a song. These fundamental frequencies can be mapped directly to piano keys.

Start-End Detector

While the previous modules interpret frequency information, this module interprets time information. It will detect when notes begin and end by looking for onsets and offsets of frequencies in our “spectrogram”. Required storage space in RAM on the FPGA is reduced by only storing note edges (begin and end) as opposed to active indicators (on or off).

Iterative Filtering

This module “windows” the input from the microphone. Each window is passed into the FFT. This method yields information in both the frequency and time domains, resulting in the desired spectrogram.

Routing & Strategy

While audio input is relatively unconstrained (any number of notes may be played in any order), use of robotic hands restricts how much of a song is playable mechanically. The Routing and Strategy module analyzes an interpreted string of notes and strategically prioritizes the most important notes in the song to play.

Music FSM: Pause, Play, and Beat Control

We implement a Moore finite state machine to control the playback of music. The finite state machine takes inputs from the play_in button on the FPGA to enable or disable playback. This includes a clock that will feed in the appropriate time scale for the interpreted music. This clock times the output from the Frequency Highlight module to then mix with the Frequency-Key Map and prepare for output to the controls block.

This FSM also includes state information as to what state the entire system is in — recording, processing, or playback.

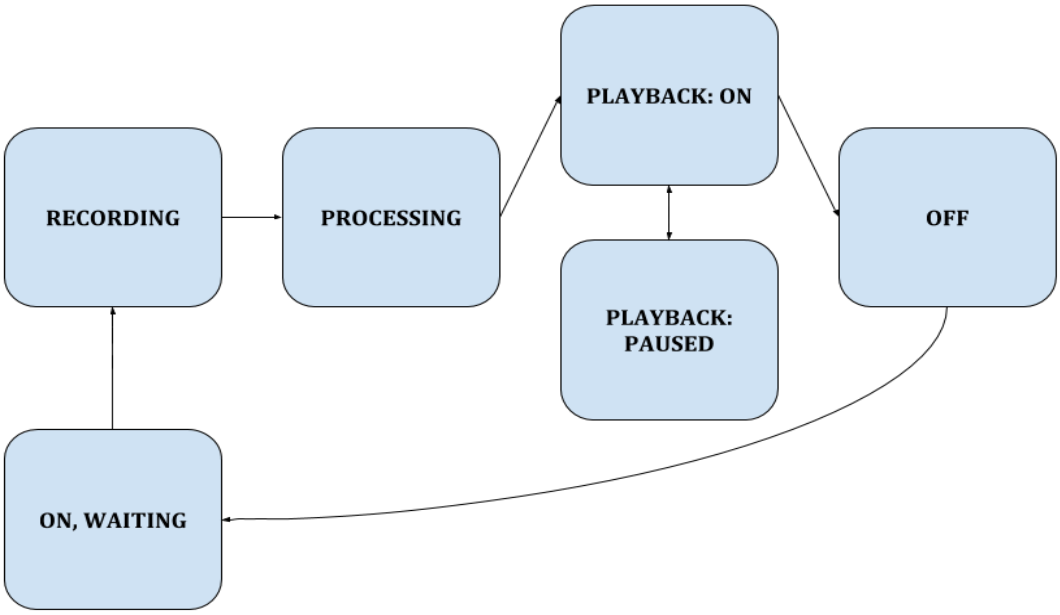


Figure 2: State Diagram

Frequency-Key Map

This map acts as an interpreter for the controls aspect, converting approximate frequency values to the note it corresponds to. Octave information is held here as well, as the Controls & Operation block may be subject to improvements, so the only information to be passed on to Synchronization & Timing is the note values instead of frequencies to be decoded.

The Music FSM will clock into the map in order to progress in the time dimension as playback occurs. The Frequency Highlight can be indexed in that time dimension, which the Frequency-Key map then mixes to create a serial input to the Controls & Operation block.

Controls & Operation

Given a map of key frequencies and their activation timelines, the final operation of the pianobot is to actuate these keys on a real anchored keyboard or piano. In order to play the keys in real time and accurately, this block focuses on the control of servo motor arrays acting as the robot's hands. Frequency maps are converted to servo actuation maps, and the robot will be able to output music by operating a completely separate device able to be played mechanically.

Synchronization & Timing

The Frequency-Key Map provides serial input in the form of notes activated in the given time scale. This module will optimize key changes to focus on actuating only the motors necessary to play the harmony, given the controls limitations we have. For example, if we are constrained in notes changed per unit time, this module is responsible for delaying the actuation of notes until the servo motors are available to do so. Finally, this module finalizes timing, taking in a beat clock from the Music FSM and ensuring the motors act optimally.

Key Serial to Parallel

This module will act as the communications output with the driver to the servo motor array. As the input has already been mapped to notes rather than frequencies, the Key Serial to Parallel module uses optimized and mapped keys from the Synchronization & Timing data stream to multiplex angular PWM outputs to the servo hand structures. This way, I/O pins can be minimized by operating the motor arrays in a similar fashion to a Seven-Segment display. The amount of servos per hand can be divided by three-four inputs to minimize wiring and increase efficiency of simultaneous motor operations.

Angle-PWM Converter

Interpreter between an angle of actuation for a given servo motor and its PWM equivalent value. Given that servo motors operate by writing angular positions with PWM, in order to control a given servo motor, the system must first control its angle of depression. The

Angle-PWM Converter will then return the duty cycle of a clock it must then operate under in order to program the motor's new angular position.

System Limitations

Note Correction

Given that Fourier Transforms can be variable in their representations of chords played by a piano vs. a band, it will be difficult to perfectly replicate any given audio input with 100% accuracy. This can be due to the introduction of harmonics that are not as well defined when audio has multiple instruments playing, each with a different timbre, and thus a different sound to the human ear. It will be a considerable challenge to perfect the system in replicating not only single-musician live play, but also in band recordings featuring a myriad of sounds and instruments (funk, dubstep, etc).

This challenge does not have a note correction interface planned, but future iterations and developments may feature a method for note correction during the playback, so that any musical errors can be corrected upon in real time without having to record again. For current planned implementation, an error such as this would be solved by simply entering the Listening state once more and reinterpreting the audio input until the mechanical output meets the operator's standards.

Instrument Interpretations

While interpreting melodies (sequences of single notes) should be relatively straightforward, this will not necessarily be true of chords. This is due to potential inharmonicity of certain instruments. Most string and wind instruments are nearly perfectly harmonic, meaning fundamental frequencies can be isolated with relative ease even in the case of chords. However, in the case of inharmonic instruments/sound sources, it will be very challenging to interpret chords. It is perhaps beyond the scope of this project, especially as for every instrument we can implement, there's always a harder one to interpret and test.

Materials

Adafruit 16-Channel PWM Servo Driver - PCA9685

This driver will act as an integrating driver for a given array of servo motors, providing the required current draw from each operational motor, as well as providing a means of multiplexing the output logic for which motors should be depressed and to which angle of depression.

MG90S Mini-Servo Motors

These Mini-Servo Motors will provide a low-weight (13.4 g), low-power solution to a means to actuate PLA fingers to press the keys. The Servos are roughly 22.5 x 12 x 35.5 mm in volume, and are easily mountable, with a stall torque of roughly 1.8 kgf•cm at 5V.

PLA 3D Printed Robotic Hand

CAD Modeled Robotic Hand structure that can constrain motors and PLA printed fingers in an expandable lattice structure for efficient actuation to press keys on the keyboard it is anchored in front of.

Sennheiser e935 Condenser Microphone

Condenser Microphone with a frequency response of 40-18,000 hz. Connected by an XLR-3 input. This microphone will be the instrument with which audio inputs will be processed and read into the FPGA as a soundwave.