UNIVERSITY OF WASHINGTON

# Music Player

Final Project

Physics 434

Reina Kato & Huong Vo

December 12, 2014

## Contents

# 1 Introduction

## 1.1 Purpose

With this software design document, we aim to describe the architecture and system design of a specific implementation of a music player. We want to focus on giving the user a visual account of a music file, along with an auditory experience.

Our music player composes of two parts: a player "piano" that allows the user to play a song and record the tune for future plays, as well as a player that takes in spreadsheets and play the inscribed music. When the user plays a note on the piano, a certain LED will light up, corresponding to the played note. The user will also see the note on the front panel, on a waveform graph. When the player is finished, a spreadsheet will be generated with all the notes played during their session.

In summary, this program is capable of outputting the sound for a given note or music file that is being played, and uses an array of LEDs for visual representation of the frequencies along with a waveform graph that shows the relative frequencies of the whole file.

# 2 System Overview

For our project, we want our program to be capable of recognizing tones and notes, and be able to distinguish that note through an array of LED. We will use the fact that each note is associated with a certain frequency. More information behind each components of the system will be given in the forthcoming sections.

## 2.1 The Physics Behind Sound

Sound, just like any other wave, is caused by a form of emitting energy through some medium – vibration through air. With the vibration, regions of low and high pressure are created and spreads outward, creating a longitudinal wave through the medium. Each wave, depending on the source, will have a different amplitude (volume) and frequency (pitch), giving us the wide variety of sound we hear everyday. The human ear is capable of detecting frequencies ranging from 20Hz to 20000Hz, just from the fluctuations in air pressure that the eardrum can feel.

Music is nothing more than a series of notes that sound pleasant when put together. We know that when any two notes whose frequencies make a 2:1 ratio, they will produce a very pleasing sound to the ear (they are also said to be separated by an octave). Some other pleasing ratios are 2:3, 3:4, and 4:5. At one point, however, the ratio of the frequencies will be too great to sound pleasant – the cutoff is at about 18:17. Why does this happen? If we were to take a look at some waveforms, the answer will be a lot clearer.

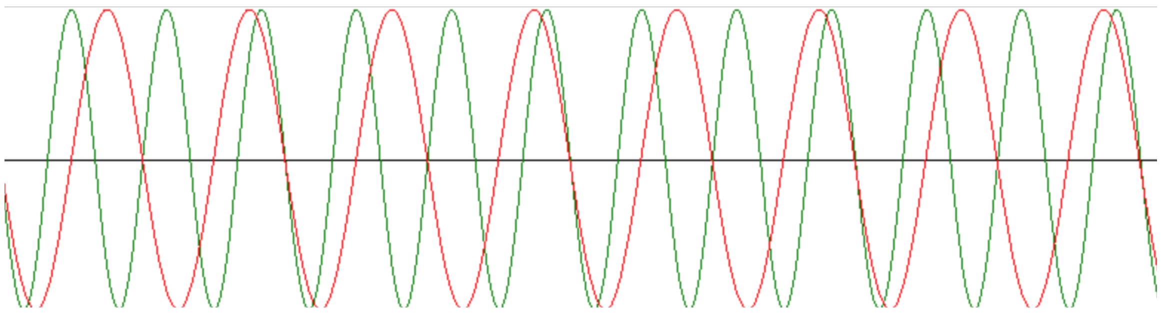


**Figure 1:** Wave representation of C and G

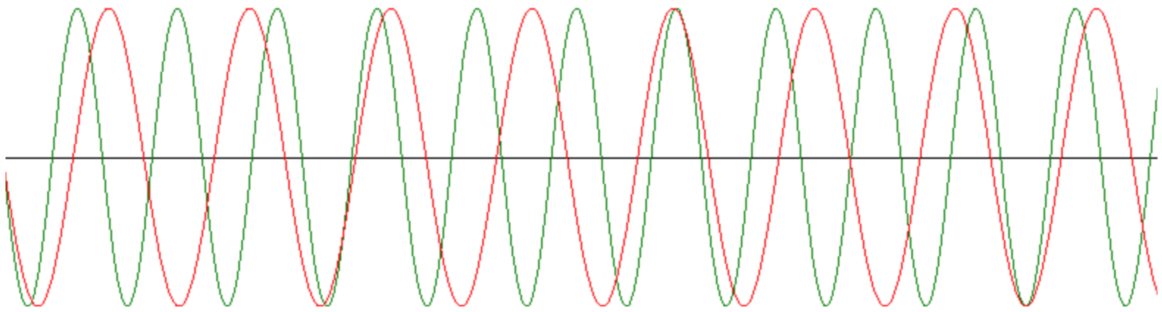


**Figure 2:** Wave representation of C and F$^{\#}$

In the first graph, we can see that there's a repeating pattern, every third peak of G will match up with every second peak of C. We can't find any such pattern in the second graph. So in order to create a harmonic combination of sounds, select notes with frequencies that coincide at regular intervals.

In music, concert-A, the note above middle-C with a frequency of 440Hz, serves as the standard tuning for musical pitch[1]We can get each successive pitch by multiplying or dividing the previous by the twelfth root of two. We can also find a note by raising this root by the number of step that note is away from concert-A. The twelfth root was found through the fact that with each octave raise, the frequency of a note is multiplied by 2. And since we know that there are twelve half-notes in an octave, we can write:

$$x^{12} = 2$$

solving for $x$ gives us approximately 1.05946. And the formula for finding any frequency relative to concert is:

$$f_n = f_0 \times x^n$$

where, $f_n$, represents the frequency we are trying to find, located n-half steps away from concert A, and $f_0$ represents the frequency of middle-A (440). This is what we are using in our program to convert from a list of pitch/note to a frequency that can be played through a speaker.

## 2.2 How Speakers Output Sound

Within a speaker, there are two main components: an inductor that will generate a magnetic field when current runs through it, a permanent magnet that is attached to a cone/membrane of some sort. In order to first generate sound, the computer will output a signal with a frequency that will create that sound. This alternating signal is ran through the inductor, and we know that as this signal changes direction, so will the magnetic field that was induced by the current. If we have a magnetic field that is directly next to the coil, it will be attracted or repealed to the coil (depending on the direction of the current). If the magnet is not fixed, it will be free to move back and forth. And if we attach this moving magnet to some membrane, the membrane will move back and forth as well, amplifying the vibration and pumping the sound waves into the air. (This is also how microphones work, but in exact opposite).

[1]Around 400 years ago, it was standard to tune keyboards (mostly harpsichords and organs) to a particular key, D-major, so that all the instruments sounded good in that key. The keyboard sounded great in this key, but particularly bad in any other unrelated keys (such as B-flat). It was decided by the composer, J.S.Bach, that keyboards should be tuned so that each notes are evenly spaced so that the keyboard will sound equally good in any key. However, this means that all stringed instruments have to allow for different tunings between each different instruments.

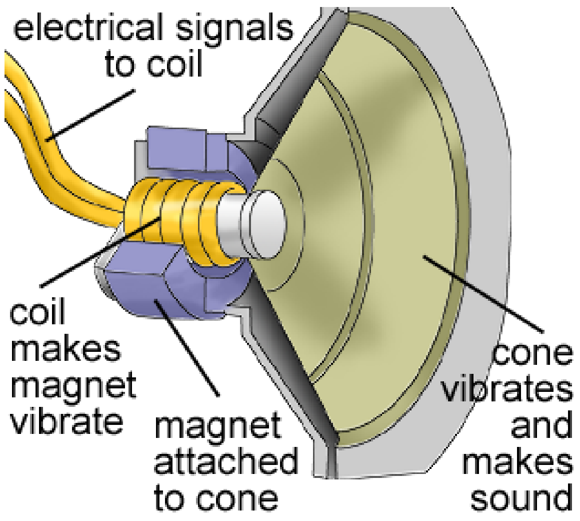

electrical signals to coil
coil makes magnet vibrate
magnet attached to cone
cone vibrates and makes sound

**Figure 3:** Internals of a Speaker

## 2.3 How LEDs Output Light

Normal, everyday materials can be divided into two sections, conductors and insulators. Conductors have electrons in the conducting band that are essentially free to move within the material (and they will move, in response to an external electric field). Insulators have electrons that are tied to their host atoms, that can be polarized due to an external field as well, but will not be free to move around. However, when an electron in an insulator receive enough energy, they can jump the energy gap and be able to conduct. Like the name implies, semi-conductor lies somewhere between a conductor and an insulator, and they achieve this through a process called doping (i.e., when foreign atoms are added to the regular crystal lattice).

There are two types of doping, n-type and p-type. With n-type doping, phosphorus or arsenic is added to the silicon in small quantities. Phosphorus and arsenic each have five outer electrons, so they?re out of place when they get into the silicon lattice. The fifth electron has nothing to bond to, so it?s free to move around (electrons have negative charge, hence the name). In p-type doping, boron or gallium is the dopant. Boron and gallium each have only three outer electrons, and when mixed into the silicon lattice, they form holes in the lattice where a silicon electron has nothing to bond to. The absence of an electron creates the effect of a positive charge (hence the name p-type). A hole happily accepts an electron from a neighbor, moving the hole over a space. N-type and p-type silicon have some very interesting behavior at the junction - this is what happens in a diode.

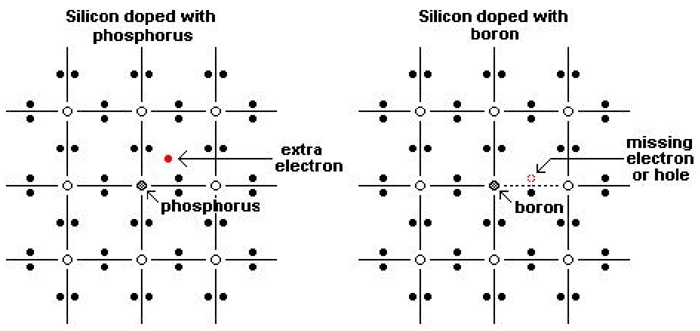


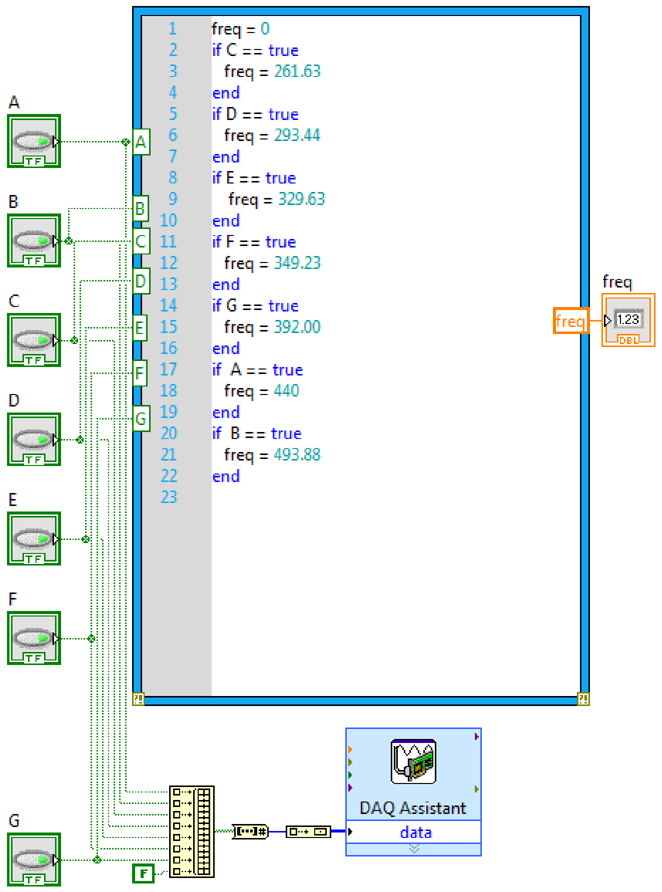
**Figure 4:** N-Type vs P-Type

When the n-type and p-type silicon is combined, the electrons and holes from each side at the interface will diffuse in order to satisfy the unpaired bonds. However, in doing so, they leave the regions on either side of the interface with a net charge, which creates what is called a depletion region, depleted of free moving electrons) inhibiting any further electron transfer unless there is an external electric field is generated. We can think of this like a voltage difference, the holes do not want to climb up the barrier and the electrons do not like falling down (the barrier is around 0.6 V). When we crank up the voltage, this effectively lowers the boundary between the electrons and the holes and the electrons start flowing. This is why diodes act like a one way current; holes and electrons start recombining at the interface, and it looks like there's current. When the electrons and holes recombine, they release energy, in the form of heat or light. The diode that emit light is called a light emitting diode. The size of the gap between the conduction band (where electrons are free to move) and the valence band (where electrons are stuck to the atom) categorize the frequency of the photon, and in return, color of the light LEDs give off.

## 3 System Architecture

### 3.1 Architectural Design

**playNote** playNote uses an array of boolean values, which checks for which note is 'on' at each time. This boolean array is converted into a number, which is then fed into a DAQ. This DAQ looks at the number and determines which individual line within a port should be on and which should be off, lighting up an LED corresponding to a specific note. This subVI will also figure out the frequency associated with the chosen note.



```
freq = 0
if C == true
   freq = 261.63
end
if D == true
   freq = 293.44
end
if E == true
    freq = 329.63
end
if F == true
   freq = 349.23
end
if G == true
   freq = 392.00
end
if  A == true
   freq = 440
end
if  B == true
   freq = 493.88
end
```

**Figure 5:** Block Diagram for playNote

Each line of Port 0 of the DAQ was wired to a resistor of value 100 Ω, so that we would not burn out the LED. This subVi is designed to play one note at one time.

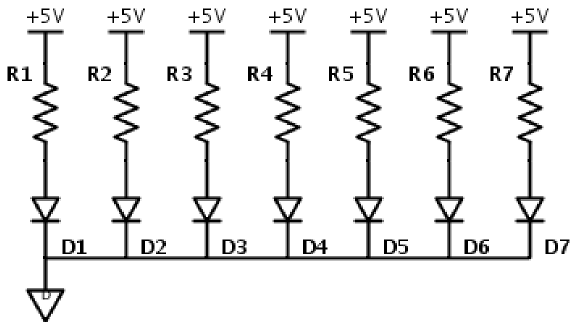


**Figure 6:** Circuit Schematic

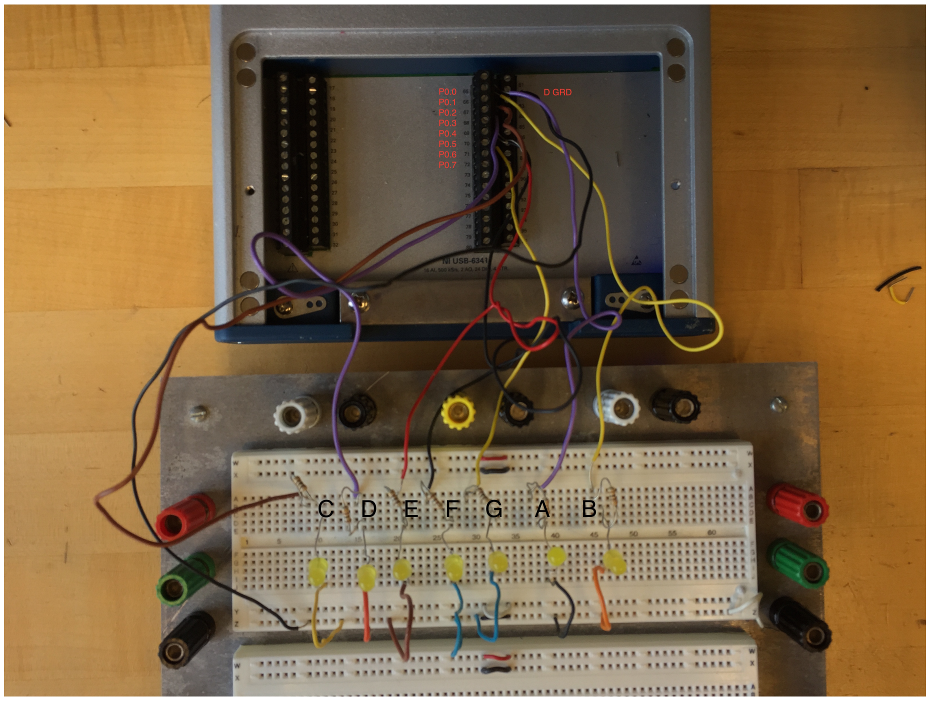


**Figure 7:** Breadboard Layout

We wired the circuit so that the first line of Port 0 correspond to the first note, C, the second line correspond to the second note, D, and etc.

**mainContainer** This is the main container containing subVIs and case structure for our piano, and our music player. This container contains two cases, one where the user is in charge of playing the song, and one in which the user can select a song from a preset list and the computer will automatically play it.

The piano, or case 1, contains 7 boolean values that is wired up to the playNote subVI, within a for-loop. Every time the player pushes a button, that translates to one iteration of the loop and the playNote subVI will light up the corresponding LED, along with returning the frequency. This frequency will be recorded by the mainContainer and read through a waveform graph and a spreadsheet when the player is done with the session. This VI is responsible for outputting the sound.

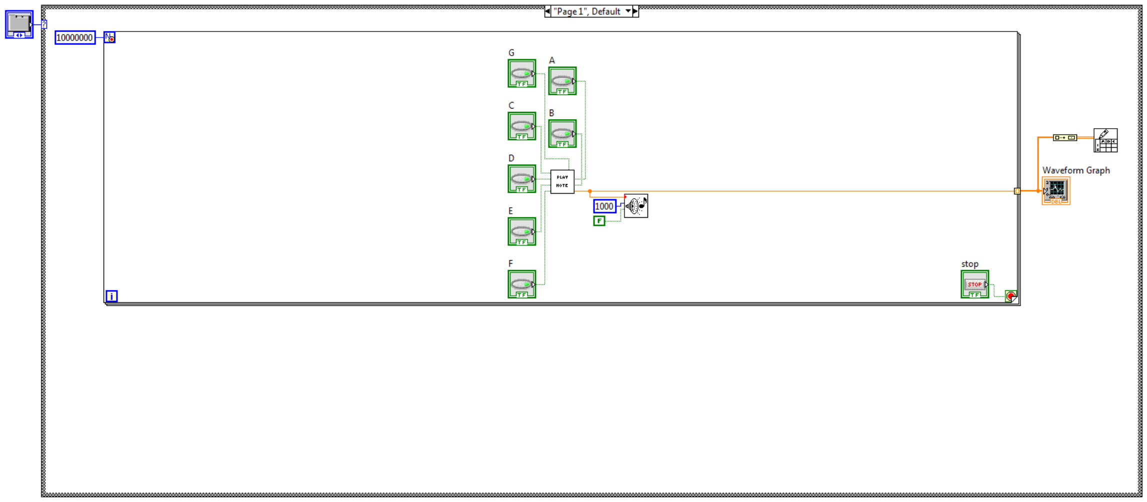
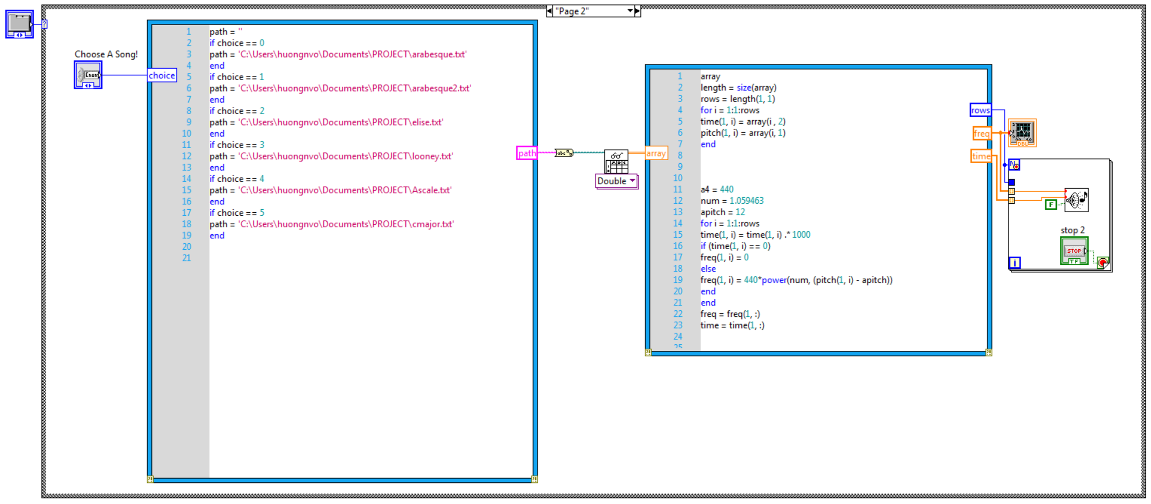


**Figure 8:** Block Diagram for mainContainer

The music player, or case 2, contains a preset list of songs that are within the folder for this main project. The user will choose one of these songs and it will be played to a speaker, as well as displayed to a waveform graph. The user will also have the option to replay the songs they prerecorded earlier. The program achieves this by reading in spreadsheets of certain values for pitch and a time that pitch should be 'on'.



```
path = ''
if choice == 0
path = 'C:\Users\huongnvo\Documents\PROJECT\arabesque.txt'
end
if choice == 1
path = 'C:\Users\huongnvo\Documents\PROJECT\arabesque2.txt'
end
if choice == 2
path = 'C:\Users\huongnvo\Documents\PROJECT\elise.txt'
end
if choice == 3
path = 'C:\Users\huongnvo\Documents\PROJECT\looney.txt'
end
if choice == 4
path = 'C:\Users\huongnvo\Documents\PROJECT\Ascale.txt'
end
if choice == 5
path = 'C:\Users\huongnvo\Documents\PROJECT\cmajor.txt'
end
```

```
array
length = size(array)
rows = length(1, 1)
for i = 1:1:rows
time(1, i) = array(i , 2)
pitch(1, i) = array(i, 1)
end



a4 = 440
num = 1.059463
apitch = 12
for i = 1:1:rows
time(1, i) = time(1, i) .* 1000
if (time(1, i) == 0)
freq(1, i) = 0
else
freq(1, i) = 440*power(num, (pitch(1, i) - apitch))
end
end
freq = freq(1, :)
time = time(1, :)
```

**Figure 9:** Block Diagram for mainContainer

## 3.2 Design Rationale

In order to simplify the project, we wanted to constrain the preset music files to pre-chosen ones that only contain one octave without any sharp notes, since we are only using 7 LEDs to represent the 7 notes in one octaves (leaving out half-notes). We wanted to do this in order to minimize the wiring time, and supplies used. Although, this task could have been easily achieved, since we have three ports. However, we didn't think it would be necessary. With that constraint, however, we wanted to show the full capabilities of the music playing section, so we chose to not wire the LEDs to the second part of the mainContainer. However, given extra time, we could have definitely pulled it off, since the playNote can take of that for us. We just didn't think it was the most important thing to concentrate on, given the time constraint.

The issue we were also concerned with was displaying the sound and the frequencies concurrently, and getting the timing to match up correctly. We were able to get the timing of the LED and the sound correctly, but we felt it wouldn't be as important to get the frequency outputs on the waveform. We were able to get all the frequencies output when the user was done running the VI.

Something we noticed was that there was a definite delay in the rate at which the LED is lit up. We think reducing the resistor values would have helped with the delay time, but we didn't want to burn out the LEDs. This was really a physical constraint that we weren't sure how to deal with. As a result, the piano cannot respond to a fast playing speed. This was another contributing reason as to why we didn't want to wire up the music player section to the LEDs, as the delay would prove to be too much for the speed at which some of the songs play.

## 4 Human Interface Design

### 4.1 Overview of User Interface

Since we are building a software that is very common in the real world, we want the user interface to be as practical and as simple as possible. Because of this, we will only implement a stop button for each case, since the user can run the VI through LabView and change the volume through the system. For the piano, there will be an array of boolean buttons with corresponding notes that the player can select. We've also implemented keyboard shortcuts, so that F1, through F7 will match up to the notes seen on screen. Each button will light up when pressed, and the corresponding LED will light up as well. There will also be waveform graphs as well as the array of LEDs that will allow the user to see physically the relationship between pitches and how the song is created.

### 4.2 Screen Images

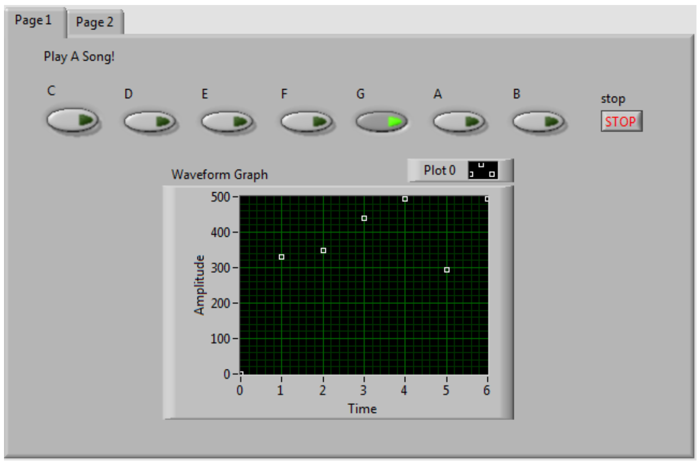


**Figure 10:** GUI for the Piano

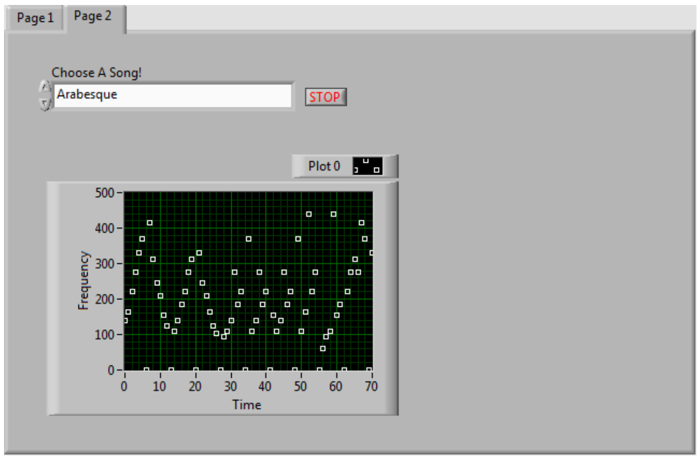


**Figure 11:** GUI for the Music Player