555 Timer Arpeggiator

Author/s: Jainam Mehta, Gene Kang

November 20, 2023

Introduction

All music is made up of 12 different tones of varying octaves. The goal of this project is to design and implement an analog machine that can take a user input to generate 3 desired notes, and sequentially output them essentially functioning as an arpeggiator. Three notes makes a chord, and a machine like this could in theory play any major or minor triad. Additionally, the user needs to know what note is being played, so there must be a mechanism that informs the person using the machine what note they are playing.

Analog music synthesizers are machines that can generate continuous audio signals from a hardware implementation. The complexity of a synthesizer can vary from it being a simple oscillator for a square or sine wave, to a modular device that uses various signal processing techniques like EQs, filters, and volume envelopes. All of these modules also have their respective hardware implementations, taking advantage of the various properties of capacitors, diodes, and transistors.

There also exists digital audio synthesis, in which through use of a computer, discrete audio signals can be generated to varying bit depths. For example, if you write a program to generate a sine wave with a frequency of 200 Hz, the resultant signal will not be continuous, but a close quantization and approximation of a sine wave using 1’s and 0’s.

Machines like these alone may seem to produce sound that may be lacking in character, but all music produced from a machine like this works in conjunction with other effects and sound processing techniques that can shape and color a tone to create beautiful melodies and feelings. The inspiration behind this project was to create a machine that would act as a foundation for synthesis. The first logical step in such a process would be the construction of a machine that could generate a pure wave. Moving forward, this implementation would bear the potential to be refined upon to create more modular segments that could do things like distort the signal, or filter it.

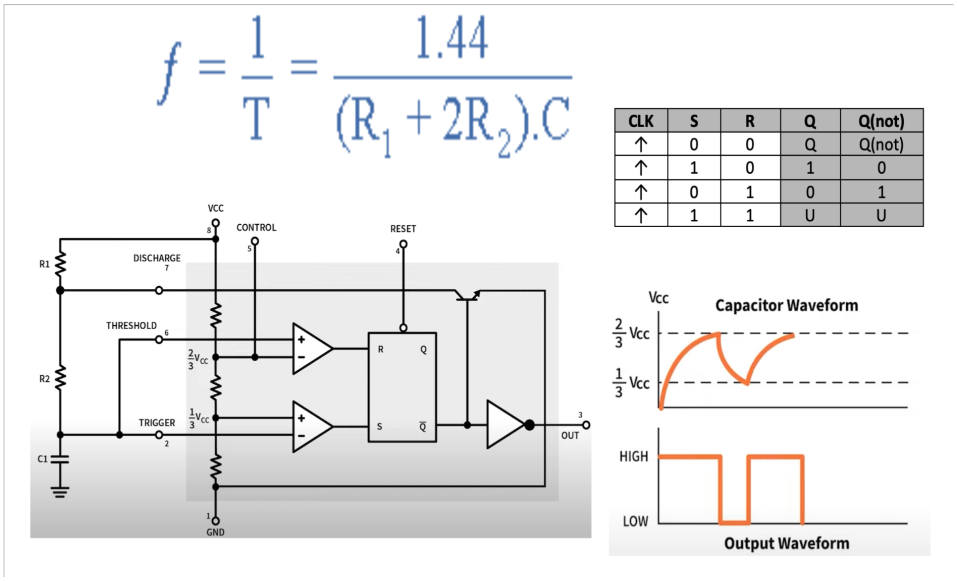
The problem statement here is to design an arpeggio synthesizer with three oscillators and a tuning module.

Research

Preliminary research included figuring out how sound was going to be generated in the first place. The arduino kits came with buzzers, which could be used to generate tone. As for the source, it was found that a VCO is capable of producing a wave of variable frequency. There are numerous types of VCO’s, with each being able to create a different type of waveform, but the most simple and suitable implementation for this case was determined to be the 555 timer circuit in its astable configuration. This is depicted in **Figure 1.1** below The circuit works essentially by repeatedly charging and discharging a capacitor using comparators and a transistor. The 555 timer will quantize the capacitor voltage and whenever the capacitor charges, the output will be a digital high, and whenever the capacitor is discharging, the output will be a digital low.

That alone is a functional circuit to produce a fixed waveform, but to give a user the ability to modulate the frequency of the waveform, would require the insertion of a potentiometer in place of R2. Keeping R1 and C constant, you would be able to change the tone of your output by altering the potentiometer dial.

Determining the values for R1 and C came from looking into what frequency corresponds to what musical note. The decision was made that remaining in a single octave would be the optimal implementation, so if a 100k potentiometer, a 100k ohm resistor for R1, and 0.15uF capacitor were to be used, we could generate frequencies from 500 Hz to over 1,000 Hz. This hits all 12 of the notes.



$$f = \frac{1}{T} = \frac{1.44}{(R_1 + 2R_2).C}$$

CLK	S	R	Q	Q(not)
↑	0	0	Q	Q(not)
↑	1	0	1	0
↑	0	1	0	1
↑	1	1	U	U

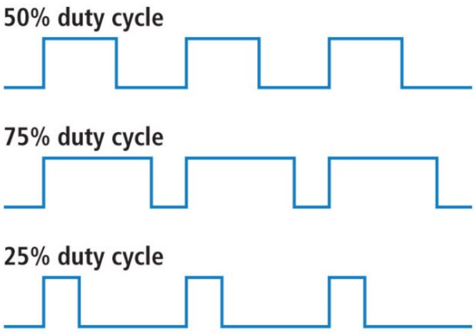
Figure 1.1

Description of Prototype

This device utilizes 3 speakers and 2 VCO's to produce the square waves received by the arduino. Additionally, an LCD panel displays the current frequency being outputted per speaker. The device is constructed using various components, including multiple breadboards, wires, a 9-volt battery, an Arduino board, 2 potentiometers for the frequency adjustments, and 3 555 timer ICs. Each speaker is controlled by a 555 timer. The device first reads the current frequency input from the VCO, and compares it to a list of predetermined frequencies in the code. Once a match is found, the device outputs the corresponding frequency mapping through the speaker and the tone note is displayed on the LCD panel. The tone notes are hard coded into the code uploaded to the Arduino. Our team's main goal is to develop a device that responds to user input and plays a 3-note sequence from that user input.

A benefit to using the arduino to generate a frequency as opposed to sending the output of the VCO directly into the speaker lies in the way alteration of the duty cycle is perceived by the listener. Two waves of the same frequency can have different duty cycles, meaning they might have the same quantity of pulses within an identical time period, but one wave may have a longer lasting pulse than another. Generally speaking, an ideal square wave has a 50% duty cycle, which is depicted in Figure 1.2. The arduino is the ideal medium to use to generate the wave, because it will read an input from what's not an ideal square wave, derive the frequency, and output a wave of that same frequency with the 50% duty cycle, which when compared to one that is of 25% or 75%, sounds much more pleasant to the ear, and bears more of the fundamental harmonic frequency.

Figure 1.2



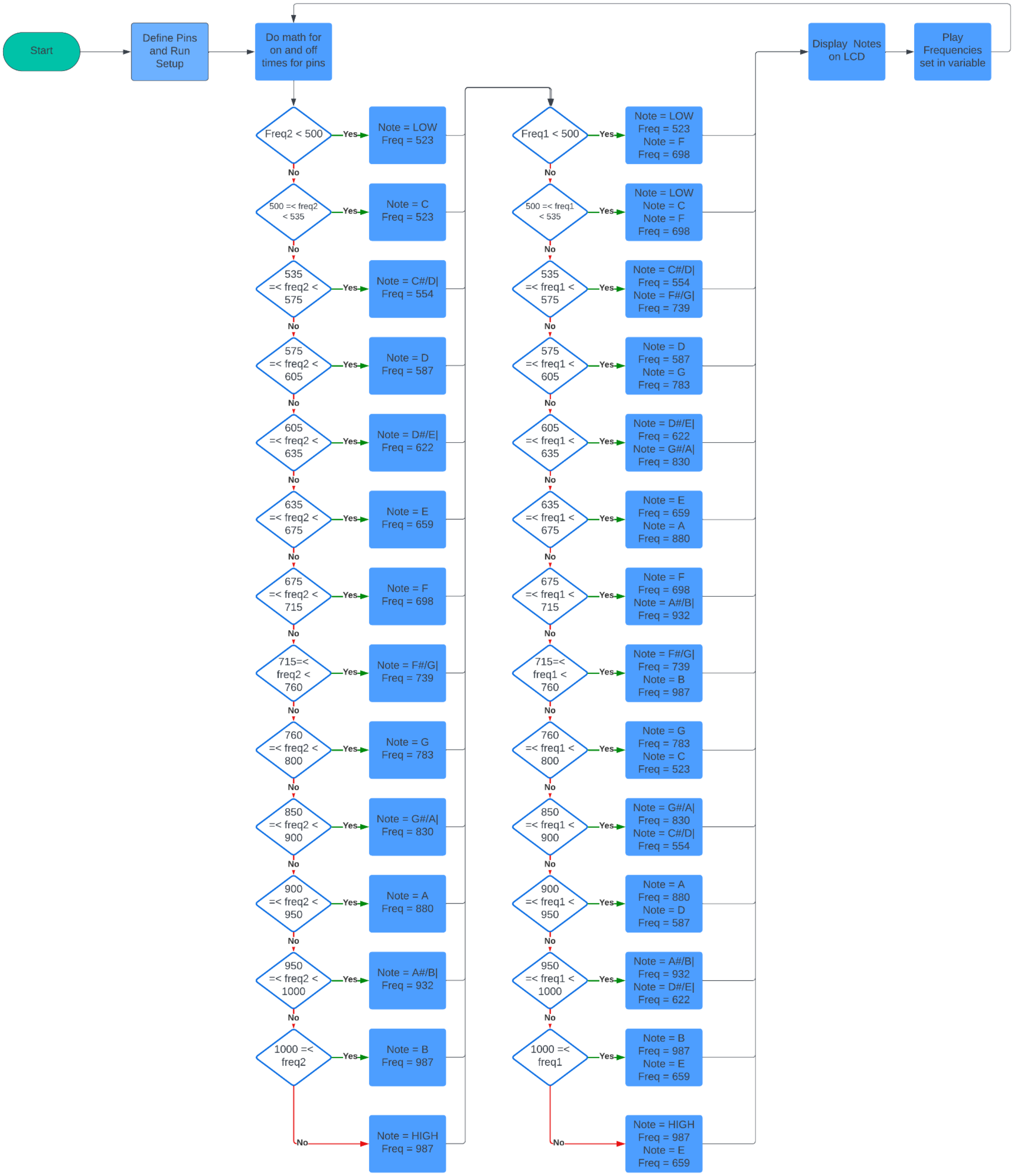
Parts:

- **(3) Piezo Buzzer**
- **(2) 0.15uF Capacitor**
- **(2) 100k Ohm Resistor**
- **(2) 10k Ohm Resistor**
- **(2) 555 Timer**
- **Arduino Uno**
- **LCD Display**
- **(2) 100k Ohm Potentiometer**
- **9 Volt Battery**

Implementation

The first step was the construction of the 555 VCO modules. Breadboards with jumper wires were used for the entire process. Initially, 3 VCOs were constructed, as the problem statement called for 3 different manipulatable notes to be available to a user. The first attempted implementation for this machine involved routing the output of the VCO straight to a speaker, as well as the arduino. The issue with this lies in the fact a note technically has a wide frequency range, so dissonance was a factor that led to the transition of using the arduino to map the VCO frequency to the buzzer frequency. This would work by using 12 different if statements to determine what note is being generated from the oscillator, and then adjusting the resultant frequency to the perfect Hz value of the note. Additionally, the if statement would also declare what note is being played on an LCD display which would also be connected to the arduino. The arduino isn't actually reading frequency, but counting the time the square wave input is in its digital high state, and adding it to the time that the square wave is in its digital low state to find the period. The code then takes the inverse of the period to determine frequency. Speakers were connected as outputs to receive the corrected frequency, which would function through use of the tone() function.

An issue was confronted when it was discovered that this tone function could only generate a tone on one speaker at a time. Additionally, due to the quality of the components in use, one of the potentiometers broke, rendering one of the three VCOs unable to function. To work around these issues, the decision to pivot from a 3 oscillator synth, to an arpeggiator was made. Additionally, due to one of the VCO's not functioning, one of the VCO's was used to not just create the note it was generating, but also the note 5 semitones up from the one it was generating. In music, this is called a perfect fourth, and will always be in tune in a major and minor triad. Thus, a three note arpeggio would be made possible using two VCO's and everything would be in correct pitch. The code to implement this was basically to use if-statements to determine the note, then sequentially output the pitches to the buzzers with a delay of 50 milliseconds.


Start
Define Pins and Run Setup
Do math for on and off times for pins
Freq2 < 500
Yes
Note = LOW Freq = 523
No
500 =< freq2 < 535
Yes
Note = C Freq = 523
No
535 =< freq2 < 575
Yes
Note = C#/D| Freq = 554
No
575 =< freq2 < 605
Yes
Note = D Freq = 587
No
605 =< freq2 < 635
Yes
Note = D#/E| Freq = 622
No
635 =< freq2 < 675
Yes
Note = E Freq = 659
No
675 =< freq2 < 715
Yes
Note = F Freq = 698
No
715=< freq2 < 760
Yes
Note = F#/G| Freq = 739
No
760 =< freq2 < 800
Yes
Note = G Freq = 783
No
850 =< freq2 < 900
Yes
Note = G#/A| Freq = 830
No
900 =< freq2 < 950
Yes
Note = A Freq = 880
No
950 =< freq2 < 1000
Yes
Note = A#/B| Freq = 932
No
1000 =< freq2
Yes
Note = B Freq = 987
No
Note = HIGH Freq = 987
Freq1 < 500
Yes
Note = LOW Freq = 523 Note = F Freq = 698
No
500 =< freq1 < 535
Yes
Note = LOW Note = C Note = F Freq = 698
No
535 =< freq1 < 575
Yes
Note = C#/D| Freq = 554 Note = F#/G| Freq = 739
No
575 =< freq1 < 605
Yes
Note = D Freq = 587 Note = G Freq = 783
No
605 =< freq1 < 635
Yes
Note = D#/E| Freq = 622 Note = G#/A| Freq = 830
No
635 =< freq1 < 675
Yes
Note = E Freq = 659 Note = A Freq = 880
No
675 =< freq1 < 715
Yes
Note = F Freq = 698 Note = A#/B| Freq = 932
No
715=< freq1 < 760
Yes
Note = F#/G| Freq = 739 Note = B Freq = 987
No
760 =< freq1 < 800
Yes
Note = G Freq = 783 Note = C Freq = 523
No
850 =< freq1 < 900
Yes
Note = G#/A| Freq = 830 Note = C#/D| Freq = 554
No
900 =< freq1 < 950
Yes
Note = A Freq = 880 Note = D Freq = 587
No
950 =< freq1 < 1000
Yes
Note = A#/B| Freq = 932 Note = D#/E| Freq = 622
No
1000 =< freq1
Yes
Note = B Freq = 987 Note = E Freq = 659
No
Note = HIGH Freq = 987 Note = E Freq = 659
Display Notes on LCD
Play Frequencies set in variable

Analysis:

Regrettably, our project encountered a number of obstacles during the development phase of our prototype, resulting in only a partial success. The root of the issue lay in our initial assumption that the Tone() function would have the ability to play multiple buzzers at once, which turned out to be proven incorrect later on. Although multiple solutions were considered, time constraints prevented any further implementation, and focus was ultimately shifted towards an alternative approach by creating an Arpeggiator that plays the three input notes in a sequence, rather than a chord machine that plays on all three speakers simultaneously, which was our initial goal for this prototype in the project. Ultimately, though a three voice synth was not attained as initially intended, what was accomplished was the successful implementation of two VCO's that generate variable frequency, a tuning mechanism that displays the note current output, and a pitch correcting arduino based program. This machine is still capable of generating music, and though is only a raw square wave generator, it bears the potential to be a foundational module in a much larger system.

Improvement:

Regarding the software aspect, with more time, the possibility of utilizing a third-party buzzer library that could play three buzzers simultaneously would be explored. In terms of the current code, there were certainly opportunities to improve it. Implementing arrays and for loops would have made the code more concise and easier to comprehend. Additionally, a third-party library dedicated to frequency calculation could have enhanced the overall quality of the code and output. To further improve the frequency estimation, a VCO that more closely hovers around a duty cycle of 50% would help the arduino not have to work with tiny pulse times.

Additionally, as for hardware, having better speakers that were more pleasant to listen to are one big opportunity for improvement. The circuit wiring was unsecured so if something like this were to be completed in the future, PCBs would be the optimal approach. They would provide an organized and reliable circuit which would be crucial in later stages of project development. However, as a prototype, what was created here served its purpose perfectly.

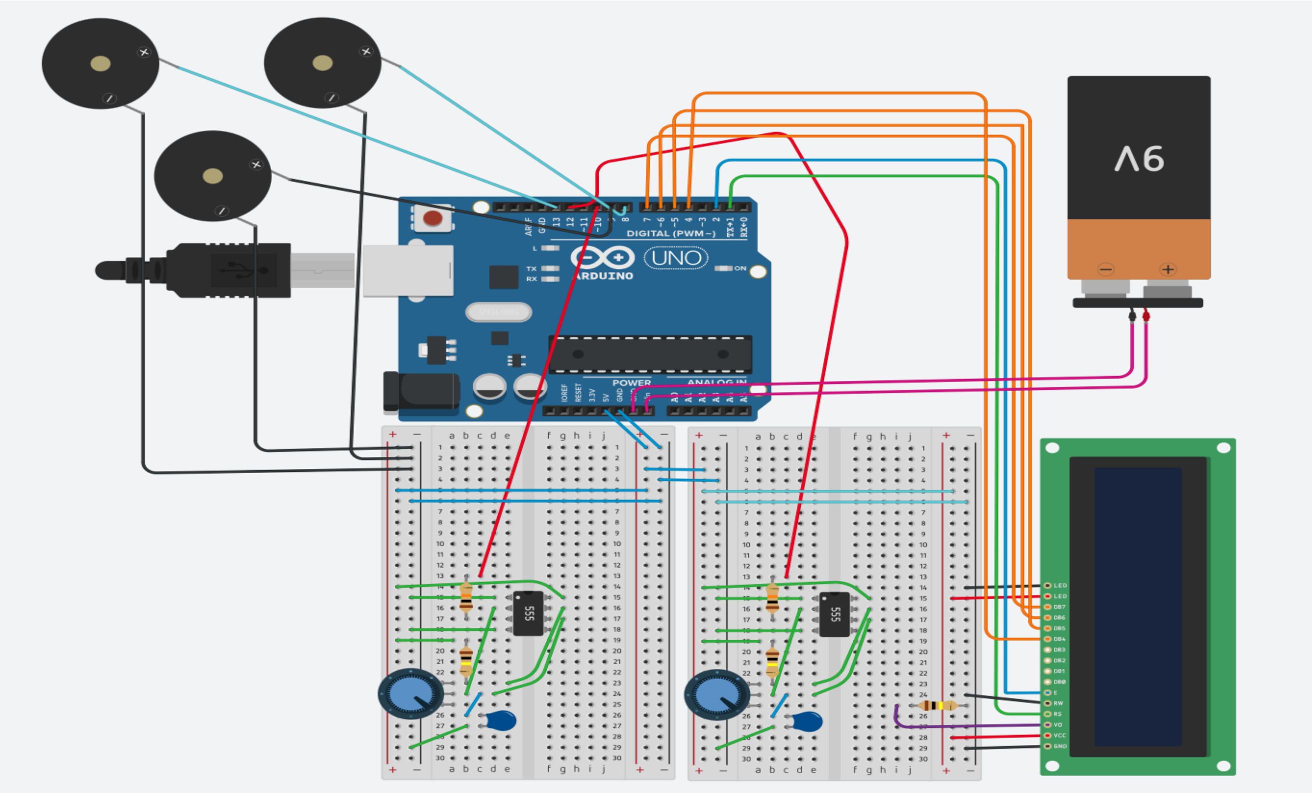
References

1. Dejan. (n.d.). *Arduino 16x2 LCD Tutorial - Everything You Need to Know.* How To Mechatronics. https://howtomechatronics.com/tutorials/arduino/lcd-tutorial/
2. CircuitBread. (2022, November 2). *Introduction to 555 Timers.* [Video]. Youtube. https://youtu.be/iwbGccGU4io?si=d6OTX6_MvDG3p02Z
3. CircuitBread. (2022, December 1). *555 Timers - Astable Multivibrator Configuration.* [Video]. Youtube. https://youtu.be/APghHcA-MOI?si=zago3YzkdkkUF5Uu

4. ambhatt. (2019, August 31). *Frequency and Duty Cycle Measurement Using Arduino.* Project Hub. https://projecthub.arduino.cc/ambhatt/frequency-and-duty-cycle-measurement-using-arduino-20fbfb
5. Music Note To Frequency Chart. (n.d.).

Appendix

Hardware Schematic:



Software Code:

```
//Library import for LCD panel
#include <LiquidCrystal.h>

//Pins that use LCD panel
LiquidCrystal lcd(1, 2, 4, 5, 6, 7);

//Pins for Frequency and Buzzers
#define pulse_ip 3
#define buzzer  9
#define pulse_ip1 10
#define buzzer1  8
#define pulse_ip2 12
#define buzzer2  13

//Declaring variables and assigning them
int ontime1, offtime1, ontime2, offtime2, freq1, period1, freq2, period2, t1, t2, t3;

const int a = 200;
const int b = 600;
String a1, a2, a3;

void setup()
{
  //Setting pin mode for pins
  pinMode(pulse_ip, OUTPUT);
  pinMode(pulse_ip1, INPUT);
  pinMode(pulse_ip2, INPUT);
  pinMode(buzzer, OUTPUT);
  pinMode(buzzer1, OUTPUT);
  pinMode(buzzer2, OUTPUT);

  //Clears the LCD monitor
  lcd.begin(16,2);
  lcd.clear();
}



void loop()
```

```
{
  //Math for frequency
  ontime1 = pulseIn(pulse_ip1, HIGH);
  offtime1 = pulseIn(pulse_ip1, LOW);
  period1 = ontime1+offtime1;
  freq1 = 1000000.0/period1;

  ontime2 = pulseIn(pulse_ip2, HIGH);
  offtime2 = pulseIn(pulse_ip2, LOW);
  period2 = ontime2+offtime2;
  freq2 = 1000000.0/period2;


//Speaker 3
if(freq2 < 500){
 a3 = "LOW";
  t3 = 523;



}
if(freq2 >= 500 && freq2 < 535){
 a3 = "C";
  t3 = 523;
}
if(freq2 >= 535 && freq2 < 575){
 a3 = "C#/D|";
  t3 = 554;

}
if (freq2 >=575 && freq2 < 605){
 a3 = "D";
  t3 = 587;
}
 if (freq2 >= 605 && freq2 < 635){
 a2 = "D#/E|";
  t3 = 622;
}
if (freq2 >= 635 && freq2 < 675){
 a3 = "E";
  t3 = 659;
```

```
}
if (freq2 >= 675 && freq2 < 715){
 a3 = "F";
  t3 = 698;
}
 if (freq2 >= 715 && freq2 < 760){
 a3 = "F#/G|";
  t3 = 739;
}
 if (freq2 >= 760 && freq2 < 800){
 a3 = "G";
 t3 = 783;
}

if (freq2 >= 800 && freq2 < 850){
 a3 = "G#/A|";
  t3 = 830;
}
if (freq2 >= 850 && freq2 < 900){
 a3 = "A";
  t3 = 880;
}
if (freq2 >= 900 && freq2 < 950){
 a2 = "A#/B|";
  t3 = 932;
}
if (freq2 >= 950 && freq2 < 1000){
 a3 = "B";
  t3 = 987;
}
if (freq2 >= 1000){
 a3 = "HIGH";
  t3 = 987;
}




//Speaker 2
if(freq1 < 500){
 a2 = "LOW";
 a1 = "F";


 t2 = 523;
 t1 = 698;
}
if(freq1 >= 500 && freq1 < 535){
 a2 = "C";
 a1 = "F";


  t2 = 523;
  t1 = 698;
}
if(freq1 >= 535 && freq1 < 575){
 a2 = "C#/D|";
 a1 = "F#/G|";


  t2 = 554;
  t1 = 739;
}
if (freq1 >=575 && freq1 < 605){
 a2 = "D";
 a1 = "G";


  t2 = 587;
  t1 = 783;
}
 if (freq1 >= 605 && freq1 < 635){
 a2 = "D#/E|";
 a1 = "G#/A|";


  t2 = 622;
  t1 = 830;
}
if (freq1 >= 635 && freq1 < 675){
 a2 = "E";
 a1 = "A";


    t2 = 659;
    t1 = 880;
}
if (freq1 >= 675 && freq1 < 715){
 a2 = "F";
 a1 = "A#/B|";


    t2 = 698;
    t1 = 932;
}
 if (freq1 >= 715 && freq1 < 760){
 a2 = "F#/G|";
 a1 = "B";


    t2 = 739;
    t1 = 987;
}
 if (freq1 >= 760 && freq1 < 800){
 a2 = "G";
 a1 = "C";


    t2 = 783;
    t1 = 523;
}
 if (freq1 >= 800 && freq1 < 850){
 a2 = "G#/A|";
 a1 = "C#/D|";


    t2 = 830;
    t1 = 554;
}
if (freq1 >= 850 && freq1 < 900){
 a2 = "A";
 a1 = "D";


    t2 = 880;
    t1 = 587;
}
if (freq1 >= 900 && freq1 < 950){
 a2 = "A#/B|";
 a1 = "D#/E|";


    t2 = 932;
    t1 = 622;
}
if (freq1 >= 950 && freq1 < 1000){
 a2 = "B";
 a1 = "E";


    t2 = 987;
    t1 = 659;
}
if (freq1 >= 1000){
 a2 = "HIGH";
 a1 = "E";


    t2 = 987;
    t1 = 659;
}



//LCD displays from variables
lcd.begin(16,2);
lcd.clear();
lcd.setCursor(0,0);
lcd.print("F:");
lcd.setCursor(2,0);
lcd.print(a1);
lcd.setCursor(9,0);
lcd.print("F:");
lcd.setCursor(11,0);
lcd.print(a2);
lcd.setCursor(0,1);
lcd.print("F:");
lcd.setCursor(2,1);
lcd.print(a3);

//Buzzer output
tone(buzzer2, t3,a);
delay(b);
noTone(buzzer2);
 tone(buzzer1, t2,a);
  delay(b);
  noTone(buzzer1);
tone(buzzer, t1,a);
delay(b);
noTone(buzzer);

tone(pulse_ip,100,b);




}
```