Sonically augmented fountains using Fast Fourier Transforms

by

Agata Gabara

November 01, 2022

This thesis is submitted in part of fulfilment of the requirements for Bachelor of Science at Middlesex University, 2022.

Abstract

Viewers appreciate musical fountains for example sonically fountains in Las Vegas. Abundance of water jets and lights gives the fountains a wonderful appearance.
What differentiate them is that their shows are synchronised with concomitant music.
Sonically augmented fountain establishes special atmosphere that can be relaxing, resembling sound of the wind chimes.

Contents

Report Structure

1. **Introduction** – Introduces aim, objectives and deliverables of thesis
2. **Problem Statement** – Provides the rationale behind the project

3. **Literature Review –** Review of existing material and to identify research methods and strategies that may be applied.
4. **Research Methodology** – Explains how the research was carried out and data collection
5. **Findings** – Discusses the data collected
6. **Requirement Specification** – Describes the various system development models and the requirements gathered.
7. **Design Specification**
8. **Evaluation –** Explanation of evaluation techniques used, results and findings.
9. **Conclusion**
10. **Bibliography –** Source of information, whitepapers, journals, articles and internet web pages used to construct this thesis.

Chapter 1
INTRODUCTION

Fountains exists in world scenarios in cities, parks and others. They may have lighting,

they can be used in multimedia shows. They are often architectural features that make public spaces more interesting and enjoyable.

This project augments fountains with addition of sound “in tune” with the sound of a fountain in a similar fashion to wind chimes. Users experience is one that is subtle and in the background.

Aim

The aim of this project is to make fountains more musical by use of specialist software and hardware.

Objectives

- to analyse sounds in Arduino,
- to generate sounds from fountain sounds,
- to find appropriate sounds to use,
- to build a functioning prototype,
- to evaluate by observation,
- to identify potential improvements.

Deliverables

- project report and evaluation,
- hardware,
- equipment to augment

Scope

Originally the project was to use the fountains at Middlesex University, unfortunately they have been turned off and are not available, therefore fountain sounds from recordings are used.

Chapter 2

PROBLEM STATEMENT

Rationale for this thesis

- fountains play important role.
- sound water is relaxing
- to enhance an urban environment for people.

Perceived Benefits

"A good musical fountain can instantly combine the visual perception of the changing water column with the listening experience of the music to create a perfect environmental artistic effect. At present, with the rapid development of computer software and hardware technology, the interactive application of new music fountain and computer is more and more extensive, and the music fountain is becoming more and more complex and more and more complete" (*Computer-Aided Design and Applications*, (2022)

Below, there is an example of fountains at the Bellagio in Las Vegas.

Bellagio Fountain - Titanic song by Celine Dion.

Dancing fountain in Las Vegas, (Musical fountain, (2022)

"Spectators greatly appreciate majestic musical fountains such as the Bellagio music fountain in Las Vegas" (…) Fountains that just make patterns with water jets have now developed into multimedia shows with music, light, and special effects. A musical fountain with synchronized water and music creates an atmosphere that can be exciting or romantic." (…) Whatever the fountain size, the choreography requires painstaking, expert programming, and their creators can only achieve this after careful analysis of the music. (*IEEE Computer Graphics and Applications,* (2009)

Chapter 3

LITERATURE REVIEW

Introduction and background.

- fountain

"The English word "fountain" derives from the Latin word *Fons*, having the meaning both of natural spring and of artificial construction built for water supply and/or decorative or symbolic purposes" (Mays, L.W.; Koutsoyiannis, D.; Angelakis. (2007)

The fountains have existed in Greece since Bronze Age. They have provided water for drinking and washing to the resident of palaces, cities, and villages. Fountains were used in some regions also as the decoration. In the ancient Egypt, the fountains were used for irrigation purpose also. Both Hellenes and Egyptians have taken into their account the contamination of surface and reinvestigate previous accomplishments.

(Hynynen, A.J.; Juuti, P.S.; Katko. (2012)

The fountains in the UK

In the United Kingdom, there are some famous fountains such as: Venus Fountain, Granary Square Fountain, Sibirica Fountain, Fountains in Trafalgar Square and others. Below, there are some pictures of them.

Fountains in Trafalgar Square (*10 Best Fountains In London, England*. (2019)

Granary Square Fountain (*10 Best Fountains In London, England*. (2019)

Soundscape approach, augmented soundscapes

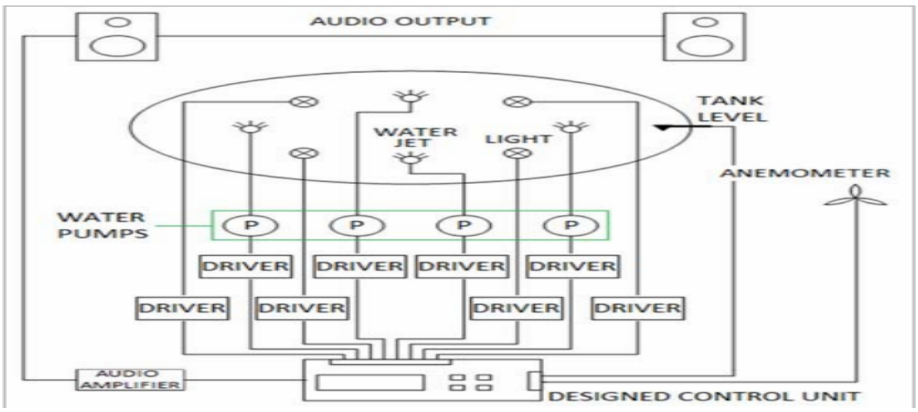
"Urban parks are able to provide multiple ecosystem services. Of major importance for people living in city centres are the social and health related benefits" (Egorov, Mudu, Braubach, & Martuzzi, 2016). "However, these benefits can be jeopardized by excessive exposure to environmental noise, negatively impacting human health (Fritschi, Brown, Kim, Schwela, & Kephalopoulos, 2011). Mainly, the abundance of technical sounds in city parks might exacerbate these services for habitancy.

The "soundscape approach" is remedy for this inconvenience. This attitude relies on

adding human-preferred sounds than mitigating unwanted ones. The scientists have made some experiments; for example, the "Musikiok "project conducted in Montreal, which aim was "to exclude music from own devices and to focus on natural sounds. Such sounds are not only highly plausible in a park environment, but are also sounds that most people enjoy" (*Landscape and Urban Planning*, (2020)

Designed control and driving system – dancing fountains in Las Vegas

In Las Vegas, it was used specialistic electronic control system, which manages the dancing fountain structure. "It drives water pumps and LED-based lights, reproduces the music files and synchronizes water and light scenarios with the music itself. This system is equipped with an mp3 reader for music reproduction and with a slot for SD memory, on which the system stores mp3 music files and the related scenarios." (…)"The controller drives, through dedicated power drivers, the water pumps and LED-based headlights in order to realize scenarios with amazing water and light plays, and at the same time it plays the music through the sound system." The system is considered as innovative because there is synchronisation between light/water play and music track.



"Block diagram of the general structure of a dancing fountain managed by the designed control and driving system." (Smart electronic system for dancing fountains control capable to create water and lighting scenarios synchronized with a music track, 2016)

The process shows in the steps as follows:

- booting system,
- fix start data,
- reporting to microcontroller by RTC,
- last access SD memory catches the file connected to programmed scenario and copy into RAM,
- microcontroller finds file and sends to mp3 decoder,
- data streaming from decoder,

- music reproduction,
- PIC's firmware reads file from RAM,
- PIC'S firmware updates the system outputs connected with water pumps and headlights controls

Architecture and acoustics

"Architecture becomes interesting when experienced via auditory senses beyond physics. Sound works with other senses of the body and influences occupants. It helps them to construct an understanding of forms, objects, and distances. The interplay between aural and visual architecture creates captivating spatial experiences." (*10 buildings with extraordinary acoustics*, (2015)

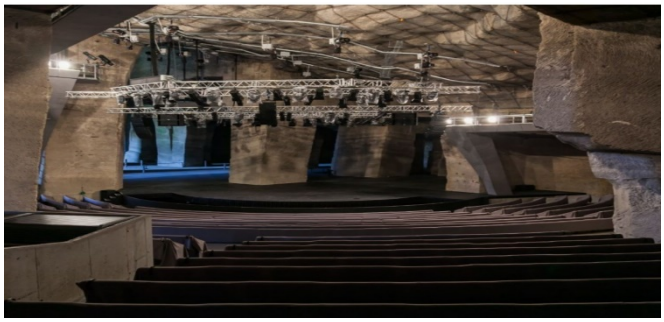
Examples:

"A capsule with a built-in sound station is suspended inside the tower. In the capsule, a team re-mixes the hum and murmur of sound from the waters of Lake Bienne, the winds of the Jura and the arteplage crowd." (*Sound Tower on arteplage Biel-Bienne*)

Klangturm, the Sound Tower, Switzerland

The Smith Centre for the Performing Arts in Las Vegas

The Hungarian Fertőrákos Cave Theatre



Sound analysis

Sound has three characteristics such as: length, pitch and intensity, which are connected with the frequency and duration of sound body vibration respectively. “Physical quantities such as amplitude and spectral distribution correspond to the three basic elements of a piece of music: melody, rhythm and harmony.” (*Computer-Aided Design and Applications*, (2022)

Basic theory of music consists of notes, length, volume, tone, pitch, name, fundamental frequency, beat, speed and rhythm. Instruments can be classified into pipes and strings relating to the way objects sound. Wind instruments are created by vibrating air in a tube, string ones are made of vibrating strings.

The presentation process of the fairy fountain is also changed with the persistent adjustment of musical emotions and the corresponding performance water type, which makes the fountain diverse.

“Music recognition methods are mainly based on short-term frequency domain analysis. Signal time domain analysis is to analyse and extract the time domain parameters of music signal” (*Computer-Aided Design and Applications*, (2022)

Analog output

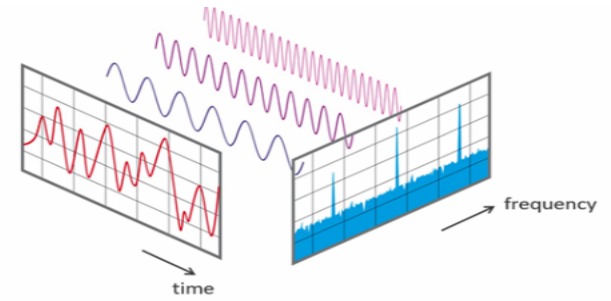
“The analog output of music fountain mainly refers to frequency converter control. Ac induction motor has three speed control methods: variable speed regulation, differential regulation and frequency control. The use of frequency conversion speed regulation can not only realize step less speed regulation of the motor, but also adjust the relationship between voltage and frequency” (*Computer-Aided Design and Applications*, (2022)

Fast Fourier Transform

Fast Fourier Transforms, the abbreviation of FFT “is a discrete Fourier transform algorithm which reduces the number of computations needed for N points from 2N (2) to

2 N lg N, where lg is the base 2- logarithm" (…) Fast Fourier transform algorithms generally fall into two classes: decimation in time, and decimation in frequency'' (Weisstein, E.W. (n.d.). *Fast Fourier Transform)*

"FFTs are used for fault analysis, quality control, and condition monitoring of machines or systems" (…) Strictly speaking, the FFT is an optimized algorithm for the implementation of the "Discrete Fourier Transformation" (DFT). A signal is sampled over a period of time and divided into its frequency components. These components are single sinusoidal oscillations at distinct frequencies each with their own amplitude and phase. This transformation is illustrated in picture 1 below. Over the time period measured, the signal contains 3 distinct dominant frequencies"



(Weisstein, E.W. (n.d.). *Fast Fourier Transform)*

MIDI

MIDI is the abbreviation of Musical Instrument Digital Interface. MIDI has been used to control audio synthetiser. Musical information has been sent between synthesizers and keyboards by MIDI protocol. In this project, it was used MIDI 1 protocol. However, it will be working with MIDI 2 also as MIDI 2 will be backwards compatible. MIDI bytes close in the interval between 0 and 255.

MIDI bytes can be present in 3 representations: decimal, hexadecimal and binary one.

Numeric range of MIDI bytes

decimal	hexadecimal	binary
0	0	0
255	FF	11111111

ccrma.stanford.edu. (n.d.). *Essentials of the MIDI protocol*. [online] Available at: https://ccrma.stanford.edu/~craig/articles/linuxmidi/misc/essenmidi.html [Accessed 19 Apr. 2023].

"MIDI commands and data are distinguished according to the most significant bit of the byte. If there is a zero in the top bit, then the byte is a data byte, and if there is a one in the top bit, then the byte is a command byte" Command bytes are divided into half. "The most significant half contains the actual MIDI command, and the second half contains the MIDI channel for which the command is for."

Division of data and commands by values

decimal	hexadecimal	binary
DATA bytes:		
0	0	00000000
...	...	...
127	7F	01111111
COMMAND bytes:		
128	80	10000000
...	...	...
255	FF	11111111

ccrma.stanford.edu. (n.d.). *Essentials of the MIDI protocol*. [online] Available at: https://ccrma.stanford.edu/~craig/articles/linuxmidi/misc/essenmidi.html [Accessed 19 Apr. 2023].

The examples of MIDI commands:

```
0x80    Note Off
0x90    Note On
0xD0    Channel Pressure
```

MIDI messages

MIDI messages = MIDI command + MIDI data parameters. 1 byte is the minimum size of the MIDI message, 3 bytes is the maximum. T

he table below presents MIDI messages that are possible in the MIDI protocol.

Command	Meaning	# parameters	param 1	param 2
0x80	Note-off	2	key	velocity
0x90	Note-on	2	key	veolcity
0xA0	Aftertouch	2	key	touch
0xB0	Continuous controller	2	controller #	controller value
0xC0	Patch change	2	instrument #	
0xD0	Channel Pressure	1	pressure	
0xE0	Pitch bend	2	lsb (7 bits)	msb (7 bits)
0xF0	(non-musical commands)			

ccrma.stanford.edu. (n.d.). *Essentials of the MIDI protocol*. [online] Available at: https://ccrma.stanford.edu/~craig/articles/linuxmidi/misc/essenmidi.html [Accessed 19 Apr. 2023].

MIDI messages - example

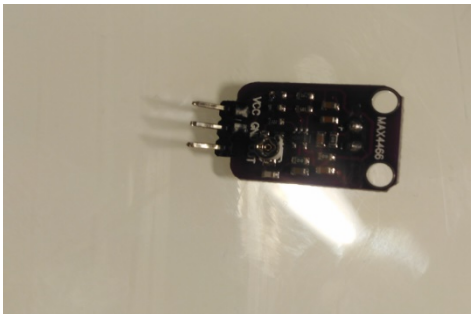
(0x90, 57, 0x45) = (hexadecimal, note number, velocity)
0x45 (hex) = 4x16 +5 =69 (dec)

To sum up, note 57 has been played with velocity 69. (MIDI is turn on)
MIDI will be turned off if 0x80 or if the note is played again with velocity set to 0.

Experiments with Arduino UNO and microphones.

By using 2 types of microphones (GY-MAX4466 and MAX9814) connected to Arduino, and using a Fast Fourier Transform, it will be identifying how water fountains produce musical sounds and enhance these sounds by adding synthesised sound of the same frequency The aim is to make fountains more musical.

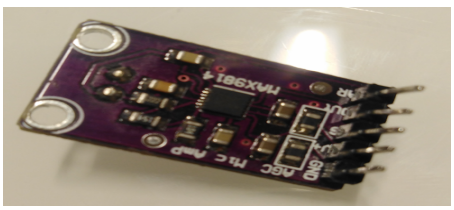
Microphone GY-MAX4466



Features:

- compact design,
- energy efficient,
- only 3 cables need to connection,
- is well protected against noise from supply of power,
- output signal is clear,
- input voltage 3-5V,
- analog signal,
- 0.8g,
- 20x14x8 mm
- electricity recording max. 0.5 mA.
- reinforcement: 25x-125x (potentiometer)

Microphone MAX9814



Features:

- "high quality microphone amplifier with automatic gain control (AGC) and low noise microphone bias" (*Max9814 microphone*)
- "The unit includes a low-noise preamplifier, a variable-gain amplifier (VGA), an output amplifier, a microphone bias circuit, a voltage generator, and an AGC control circuit. The low-noise preamplifier has a fixed gain of 12dB, while the VGA gain automatically adjusts from -20dB to 0dB." (*Max9814 microphone*)

- gains: 8, 18, 28 dB,
- 26x14x10 mm,
- only 3 cables need to connection.

Chapter 4

DESIGN SPECIFICATION

The following list presents the list of components such as: hardware, software, musical scale, functionality which have been used in this project.

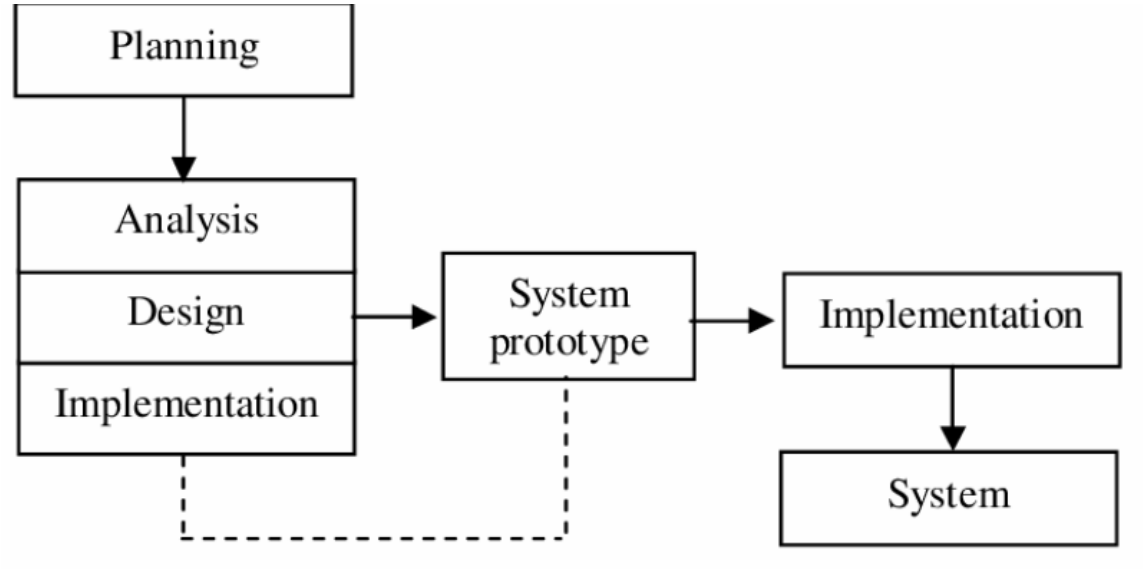
Hardware: Arduino UNO R3, breadboard, MIDI, Yamaha speakers, HUWAWEI phone, DELL Latitude E7240, jump wires, resistors, microphone MAX9814, microphone GY-MAX 4466,
Software: Arduino IDE, Synthedit 1.4 64 bit, MS Office, Windows 10,
Musical scale: mini pentatonic scale,
Functionality: full functionality (appropriate Arduino libraries must be chosen)

The system needs the microphone, and it will analyse the range of frequencies. It will be able to identify individual notes connected to synthetiser.

Chapter 5

RESEARCH METHODOLOGIES

I have used prototype approach in this project.



Anon, (n.d.). Available at: https://www.researchgate.net/publication/268376151_WEB_BASED_INTELLIGENT_APPOINTMENT_SYSTEM.

"Prototyping is the process of designing a mock-up of a product or process ahead of creating a final design" (www.twi-global.com. (n.d.). *What is Prototyping? (A Complete Guide)*. [online] Available at: https://www.twi-global.com/technical-knowledge/faqs/what-is-prototyping.

"Prototype models (…) help understand and test the functionalities and feasibility of various products" Anon, (n.d.). *What Is A Prototype Model? (Methodology, Types And Uses)*. [online] Available at: https://in.indeed.com/career-advice/career-development/what-is-prototype-model [Accessed 23 Apr. 19AD].

The aim of this dissertation is to create the product which analyses sound of fountains, identifies specific frequencies associated with notes in a given music scale.

Step 1 Using FFT function with a hardware to identify frequencies (mini pentatonic scale)

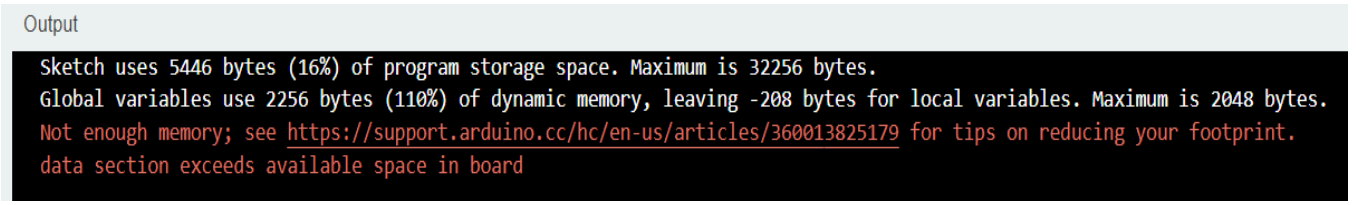
The Hamming window will be selected because it should improve the frequency analysis compared to the rectangle window. During frequencies' analysis some abnormalities can occur.

The impact on results deformation can have:

- The low quality of microphone in telephone (using Signal Generator app to generate appropriate frequencies sounds),
- Changes in output voltage,

- Preamplifier's contact to MAX9814.
- Compression,
- The capacity of memory ARDUINO UNO R3,
- Harmonics phenomena.

To avoid distortion of results some steps can be taken, for example instead of using low quality microphone in phone, Signal generator app can be run on desktop also.
The number of samples taken in Arduino code should not excess 128. If the number exceed, the communique like this can occur.



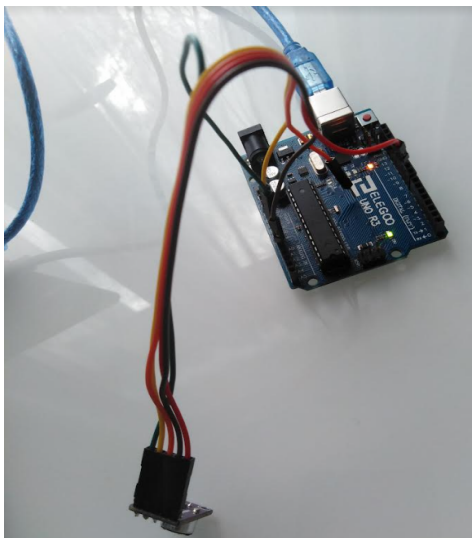
Own elaboration.

With the reference to a mini pentatonic scale's, the notes considering shows the table below:

Frequency [Hz]	*Note*
146.83	D3
196	G3
220	A3
233.08	A#3
261.63	C4
277.18	C#4
293.66	D4
311.13	D#4
329.63	E4
392	G4
415.3	G#4
440	A4

Own elaboration, based on: Michigan Technological University (2019). *Frequencies of Musical Notes*. [online] Mtu.edu. Available at: https://pages.mtu.edu/~suits/notefreqs.html.

Let's beginning with connecting microphone MAX9814 to ARDUINO UNO R3 and implementing the appropriate Arduino code.



Own elaboration.

Arduino UNO R3 has been connected to the microphone MAX9814 as follows:

A0 -> OUT

GND -> GND

VDD -> 5V

On the computer, the exact code has been run in Arduino IDE application. The code has been downloaded from: www.arduinolibraries.info. (n.d.). *arduinoFFT*. [online] Available at: https://www.arduinolibraries.info/libraries/arduino-fft [Accessed 27 Mar. 2023].

The library called arduinoFFT.h has been used in this case. The process has repeated every 500 seconds (delay 500), 115200 baud (bits per second) refer to data rate for data transmission.

The last part of the code refers to FFT. `Serial.print((i * 1.0 * SAMPLING_FREQUENCY) / SAMPLES, 1);` and `Serial.print(" ");` allows to notice which frequencies has which amplitudes.

Please look in appendix A to see the code which was used. The code has been run 21 times (number of frequencies in the table)

The comparison below illustrates frequencies from mini pentatonic scale and relevant frequencies for which the highest value has been received.

Frequency [Hz]	*Note*	*Frequency for which the highest value has been reachedd [Hz]*	*Difference [Hz]*
146.83	D3	148.4	-1.57
196	G3	195.3	0.7
220	A3	218.8	1.2
233.08	A#3	234.4	-1.32
261.63	C4	265.6	-3.97
277.18	C#4	281.2	-4.02
293.66	D4	296.9	-3.24
311.13	D#4	312.5	-1.37
329.63	E4	328.1	1.53
392	G4	398.4	-6.4
415.3	G#4	421.9	-6.6
440	A4	445.3	-5.3

Own elaboration.

The peak values mentioned in the table will be compared to the peak values received after running modified Arduino code. The code broadens of 2 lines:

- Serial.print("peak = "),
- Serial.println(peak);.

The code after modifications looks as mentioned in appendix B.

The factsheet hereafter presents the peak values for the respective frequencies. Red colour highlights that in the case of 196 Hz the peak values received in Serial Monitor in Arduino IDE differentiate from each other significantly because of the phenomena of harmonics. Due to the abnormality, the frequency 196 Hz will be skipped in further research.

Fequency [Hz]	**Peak(s) [Hz]**
146.83	148.49
196	280.25, 47.96, 44.5, 80.95, 146.02
220	222.60, 222.84, 222.48
233.08	236.59, 236.43, 236.55
261.63	265.89, 265.60, 266
277.18	281.87, 282.36, 281.80
293.66	298.37, 298.46, 298.30
311.13	315.37, 315.67, 315.63
329.63	334.35, 334.82, 334.19
392	397.82, 397.98, 398.07
415.3	421.67, 421.69
440	447.11, 447.08

Own elaboration.

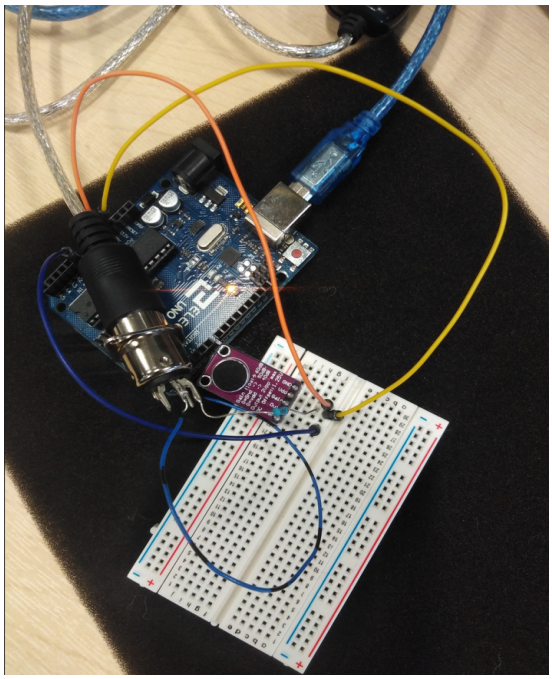
Step 2 Playing the notes on the phone and observing if MIDI generates the notes.

MIDI note on message are as follows:

1) first value – play a note,

2) second value – note,

3) third value – how loud the sound is.



Own elaboration.

Arduino UNO R3 has been connected to the microphone MAX9814 and MIDI as follows:

Microphone MAX9814

GND-> GND (orange cable),

VDD ->5V (yellow cable),

GAIN->A0 (blue cable),

MIDI

Resistor cable connected on the breadboard and second cable connected to A2 pin on digital side of Arduino UNO.

This time the code will be enriched of conditions. The condition has been created according to the formula: if (peak value == peak value from the table (step 1) referring to the appropriate note) { // appropriate note value The conditions will be created for: A3, C4, D4, E4, G4 and A4. Also, the delay time has been changed from 1000 to 10.

```
if (peak == 222.60 ) {  // A3
noteOn(0x90, 57, 0x45);
delay(1000);
noteOn(0x90,57, 0x00);
```

```
}
if (peak == 265.89) {  // C4
noteOn(0x90, 60, 0x45);
delay(1000);
noteOn(0x90,60, 0x00);
}
if (peak == 298.37) {  // D4
noteOn(0x90, 62, 0x45);
delay(1000);
noteOn(0x90,62, 0x00);
}
if (peak == 334.35) {  // E4

noteOn(0x90, 64, 0x45);
delay(1000);
noteOn(0x90,64, 0x00);
}
if (peak == 397.82) { // G4
noteOn(0x90,67, 0x45);
delay(1000);
noteOn(0x90,67, 0x00);
}
if (peak == 447.11) { // A4
noteOn(0x90,69, 0x45);
delay(1000);
noteOn(0x90,69, 0x00);
}
```

The code after modifications – appendix C.

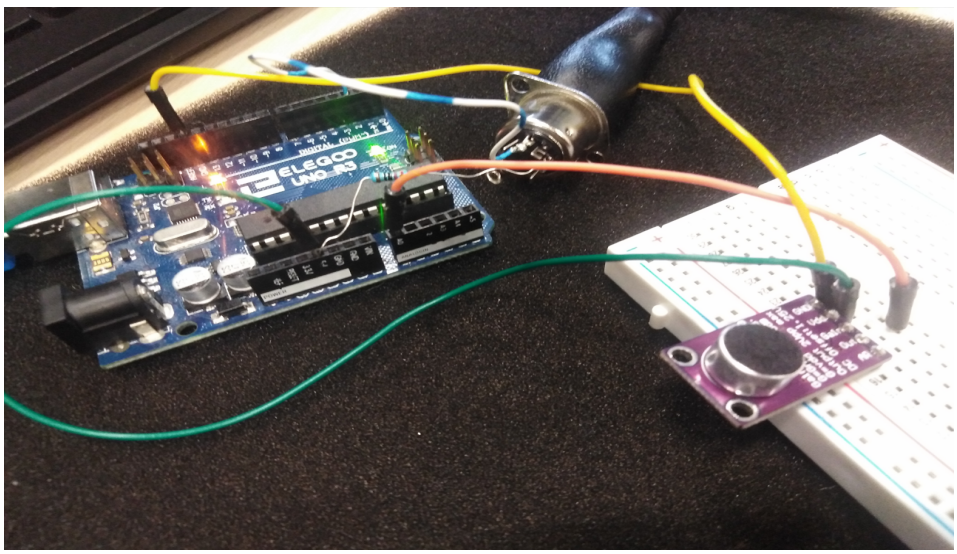
Library called "SoftwareSerialTX.h" (multi-instance software serial library for Arduino/Wiring Interrupt-driven receive and other improvements by ladyada http://ladyada.net
(Stoffregen, P. (2023). *PaulStoffregen/SoftwareSerial*. [online] GitHub. Available at: https://github.com/PaulStoffregen/SoftwareSerial/blob/master/SoftwareSerial.h)

has been implemented instead of "SoftwareSerial.h" because there was a problem with receiving playing notes; it has not worked continuously. Possibly, it was a conflict between Software Serial library and FFT library. It takes around two weeks to diagnose that problem has been caused by the library not other components. Step three presents the results. For notes such as: A3, C4, D4, E4, A4, A#4 half of the frequency is octave below. Formula for condition looks as follows: `if (peak > x && peak < y)`, where x and y creates the range of playing note; x is the lower value, y is upper value. Both values have been chosen by observing "peak" values for appropriate notes in Serial Monitor.

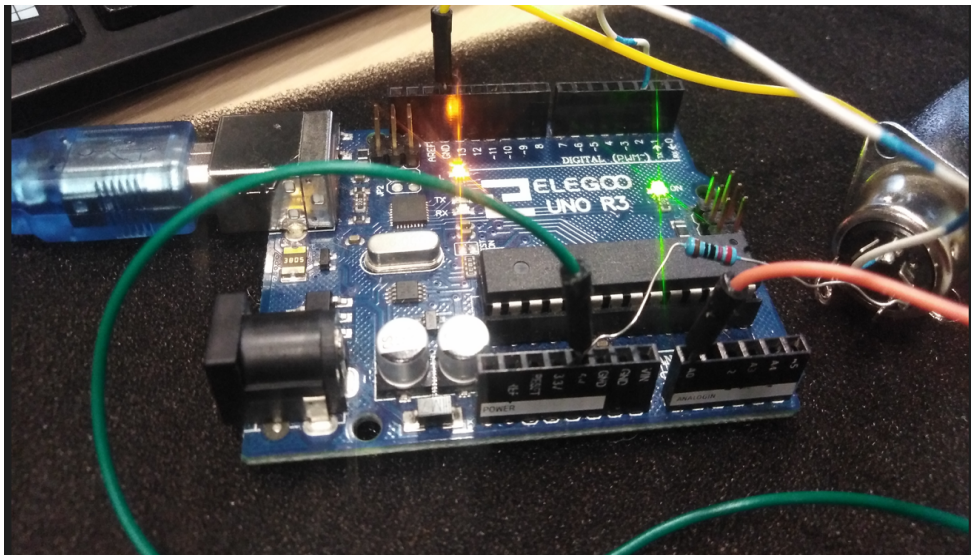
Step 3 Sound improvement by adding effects such as: Reverb Zita and Reverb DH.
Components: Synthedit, MIDI, Arduino UNO R3 and MAX9814 and breadboard.
Reverb Zita and Reverb DH have been connected to VCA and IO Mod.

Arduino UNO R3 has been connected to the microphone MAX9814 by breadboard as follows:
A0 -> OUT (orange)
GND -> GND (yellow)
5V -> VDD (green cable)

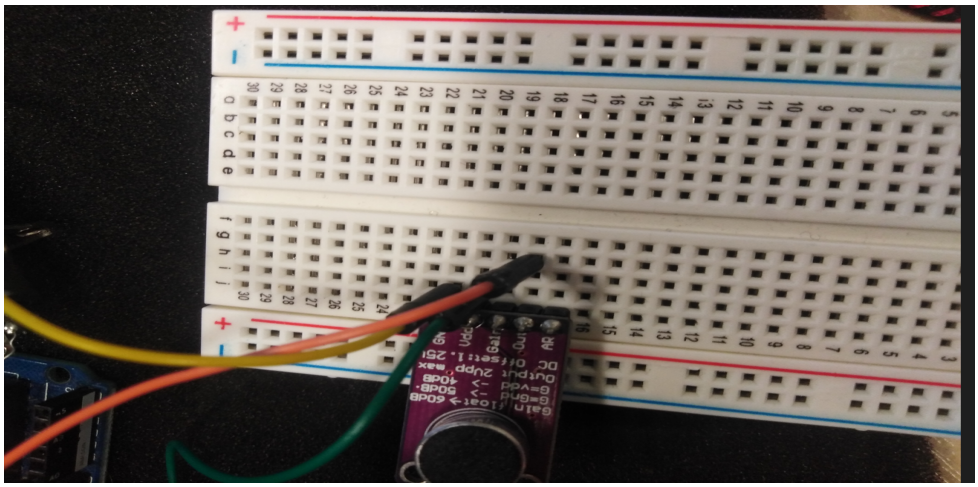
MIDI has been connected to 2 pins as follows:
resistor-> 5V
longer cable (white/blue cable) -> A2



Own elaboration.



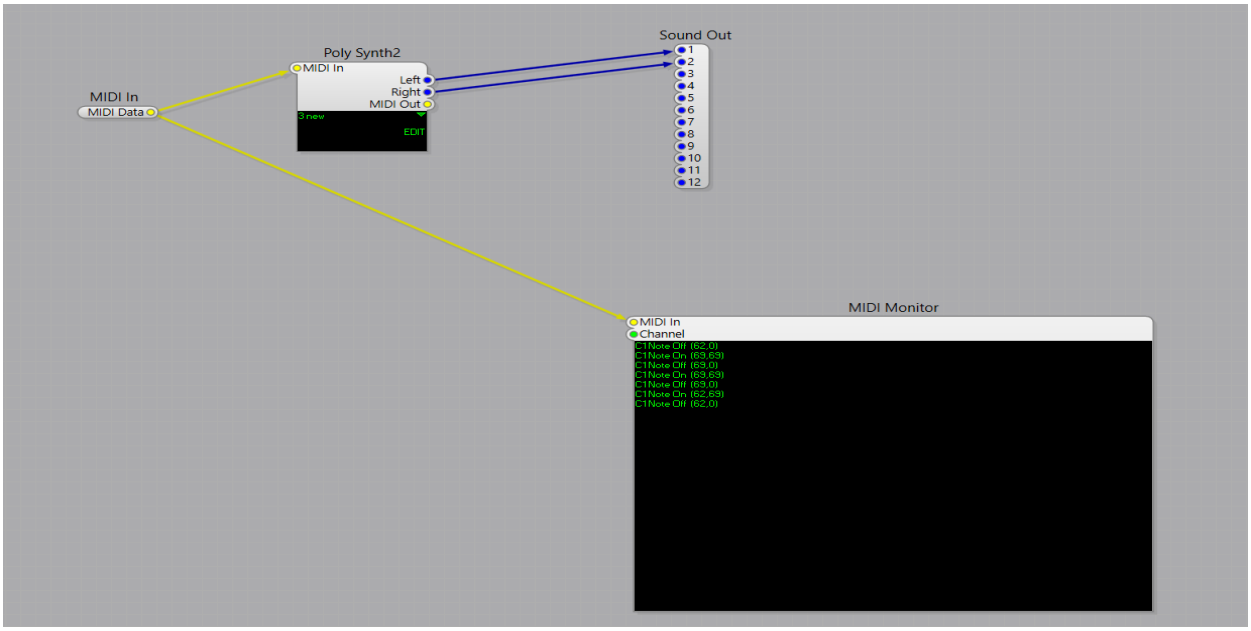
Own elaboration.



Own elaboration.

The code, which has been used presents appendix D.

In Synthedit situation looks as below. MIDI IN (MIDI Data) is connected with Poly Synth2 and MIDI Monitor. From PolySynth2 two arrows (left and right) has led to Sound Out (1 and 2) On MIDI Monitor the playing notes can be observed.

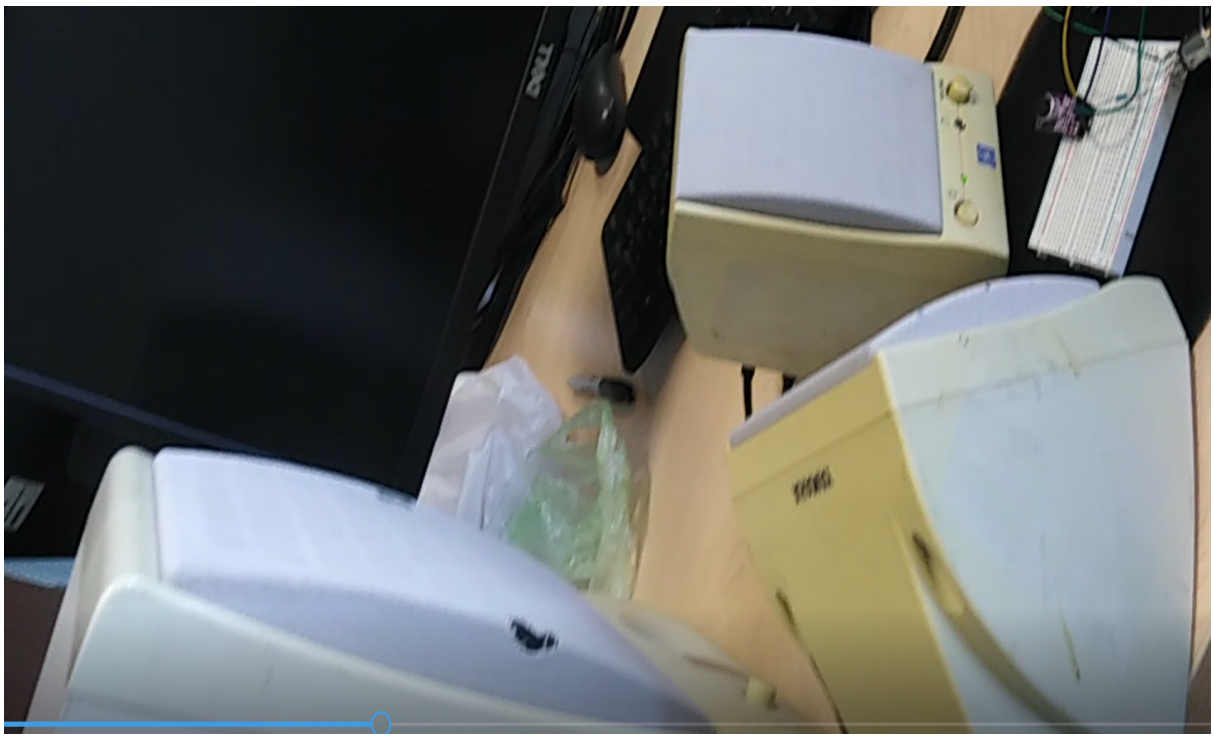


Own elaboration.

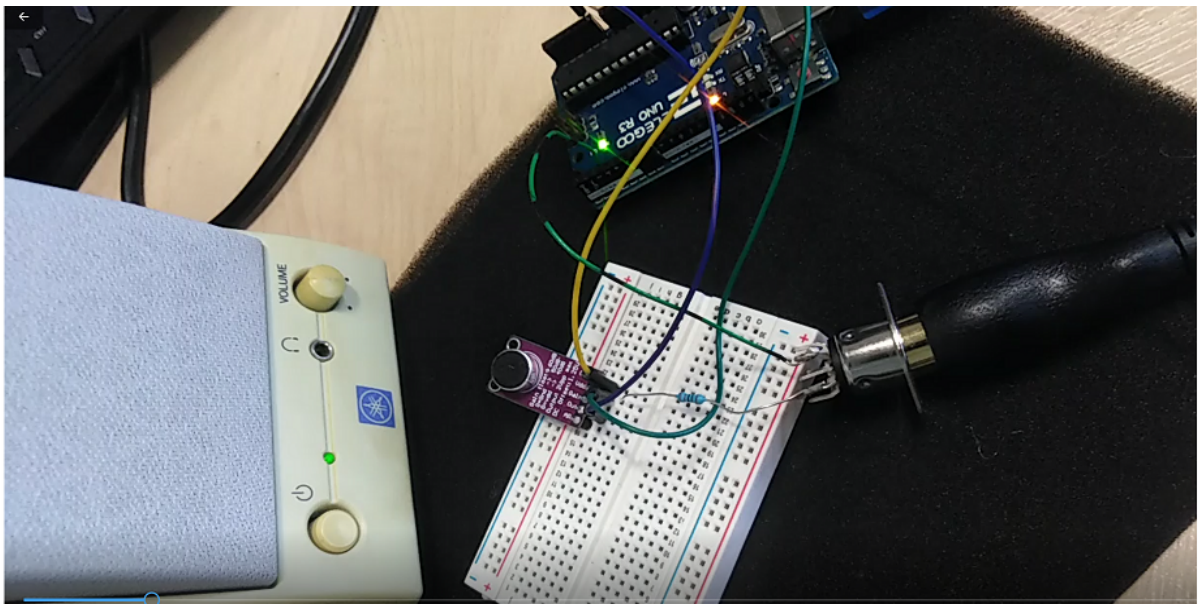
The structure has been presented in appendix F.

I have chosen from the menu " Reverb Zita" and "Reverb DH". "Reverb makes everything sound better. Especially computer-generated sound, which usually sounds a bit cold in digital, needs some reverb to make it appear more natural and organic" (Christensen, P.J. (2023). *ZITA-REV1*) and (www.youtube.com. (n.d.). *SynthEdit Tutorial 16 - Adding a Delay Effect*. [online] Available at: https://www.youtube.com/watch?v=YNhS_ftRlOk)
After this activity, the sound has better quality and is similar to a wind chimes sound.

Step 4 Adding Yamaha speakers to MIDI, ARDUINO UNO R3 and MAX9814.
The Arduino code and Synthedit structure are the same like in step 3.
The speakers set has been connected to the telephone where water sound was playing.



Yamaha speakers set, own elaboration.



Yamaha speaker set, MIDI, MAX9814, Arduino UNO R3, own elaboration.

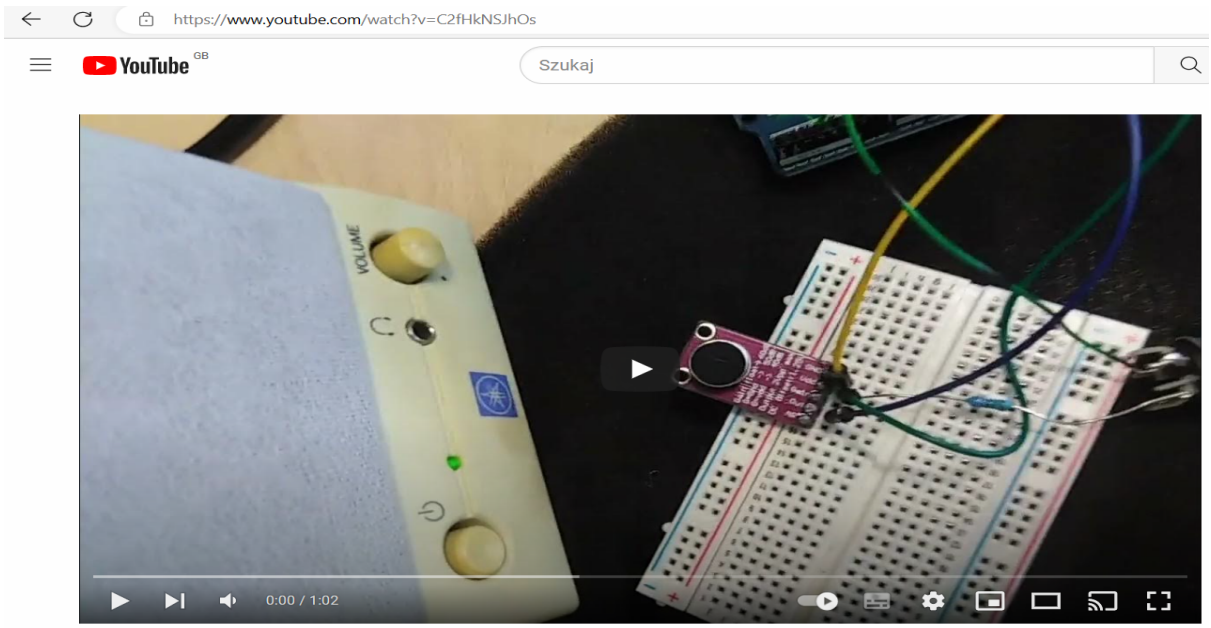
The melody comes from the YouTube, it is entitled “Bamboo water fountain”.

BAMBOO WATER FOUNTAIN | Relax & Get Your Zen On | White Noise | Tinnitus Relief. (2014). *YouTube*. Available at: https://www.youtube.com/watch?v=aJaZc4E8Y4U [Accessed 2 Mar. 2020].

BAMBOO WATER FOUNTAIN | Relax & Get Your Zen On | White Noise

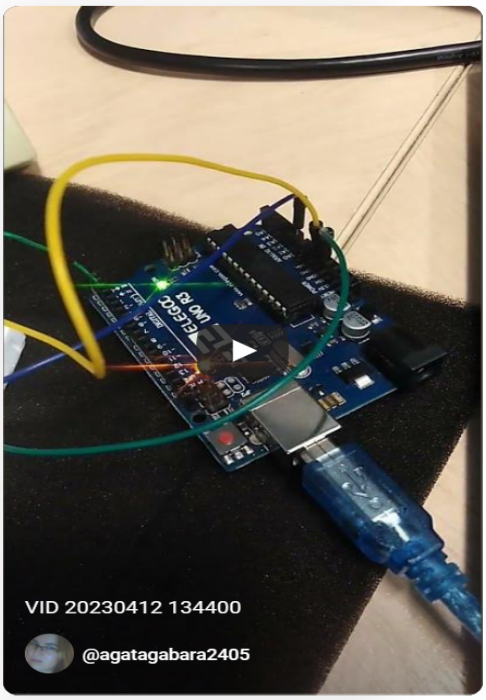
Own elaboration.

On YouTube there are two short videos presenting received sound. The links are below.



VID 20230412 134134

https://www.youtube.com/watch?v=C2fHkNSJhOs



https://www.youtube.com/shorts/tqo_rKVEjPg

Using the speaker, it was easier to notice echo. The note coming from the speaker was better quality than playing from the phone only. Still there is an issue with playing the same note the same time. It can be caused by air conditioning, phone limitations and other factors which are existing in the uni lab. The remedy for this can be small code change.

Reasons for Chosen Techniques

- simplicity,
- availability,
- price,

Chapter 7

FINDINGS

Interpretation of Results

There are some factors which impact on the results, for example: phone limitations, changes in output voltage, preamplifier's contact to MAX9814, compression, the capacity of memory ARDUINO UNO R3, harmonics, air -conditioning and others. To

avoid distortion of results some steps can be taken, for example instead of using low quality microphone in phone, Signal generator app can be run on desktop also, the air-conditioning should be switched off, the number of samples in Arduino code should not big too big (128 is ok).

In the case of minor pentatonic scale, not all the notes should not be considered. For example, because harmonics exists in some cases. Considered should be notes such as: A3, C4, D4, E4, G4 and A4.

Not without significance is also the choice of the library used in Arduino. For example, library “SoftwareSerial.h” cant be placed together with library SoftwareSerialTX.h".

For sound improvement in Synthedit were used effect such as: Riverb Zita and Riverb DF.

Plugin speakers to allows to get better quality sound and allows to hear echo.

While playing a note, the system play it back via MIDI.

Chapter 8

EVALUATION

The selected notes from the mini pentatonic scale, which have been considered in the project shows the illustration below.

Frequency [Hz]	*Note*
146.83	D3
196	G3
220	A3
233.08	A#3
261.63	C4
277.18	C#4
293.66	D4
311.13	D#4
329.63	E4
392	G4
415.3	G#4
440	A4

Firstly, the microphone MAX9814 has been connected to Arduino UNO R3 and the code from appendix A was implemented. The table below present received results.

Frequency [Hz]	*Note*	*Frequency for which the highest value has been reachedd [Hz]*	*Difference [Hz]*
146.83	D3	148.4	-1.57
196	G3	195.3	0.7
220	A3	218.8	1.2
233.08	A#3	234.4	-1.32
261.63	C4	265.6	-3.97
277.18	C#4	281.2	-4.02
293.66	D4	296.9	-3.24
311.13	D#4	312.5	-1.37
329.63	E4	328.1	1.53
392	G4	398.4	-6.4
415.3	G#4	421.9	-6.6
440	A4	445.3	-5.3

For example, for the note G3 (196 Hz), the highest value was 195.3 Hz.

Secondly, the code from Appendix C has been used to print the peak values for chosen notes from the mini pentatonic scale. For 196 Hz received peak values differ from each other significantly. The factsheet hereafter presents the peak values for the respective frequencies.

Fequency [Hz]	**Peak(s) [Hz]**
146.83	148.49
196	**280.25, 47.96, 44.5, 80.95, 146.02**
220	222.60, 222.84, 222.48
233.08	236.59, 236.43, 236.55
261.63	265.89, 265.60, 266
277.18	281.87, 282.36, 281.80
293.66	298.37, 298.46, 298.30
311.13	315.37, 315.67, 315.63
329.63	334.35, 334.82, 334.19
392	397.82, 397.98, 398.07
415.3	421.67, 421.69
440	447.11, 447.08

Thirdly, playing the notes on the phone and observing if MIDI generates the notes. Appendix D shows the code which was used.

Next, the special effects such as: Reverb Zita and Reverb DH have been selected in Synthedit to make the sound more natural.

At the end, Yamaha speakers have been connected to MIDI, ARDUINO UNO R3 and MAX9814. The Arduino code and Synthedit structure are the same like in step above. The speakers set has been connected to the telephone where water sound was playing. Using speakers allow to hear echo. It has appeared the issue with playing the same note the same time. It can be caused by air-conditioning, phone limitations and other factors.

Chapter 9

DISCUSSION

Problems

Libraries

There was a contradiction between 2 libraries such as: “SoftwareSerial.h” and “arduinoFFT.h". Library “SoftwareSerial.h” has been replaced by “SoftwareSerialTX.h" and the problem with receiving playing notes has disappeared.

Harmonics phenomena

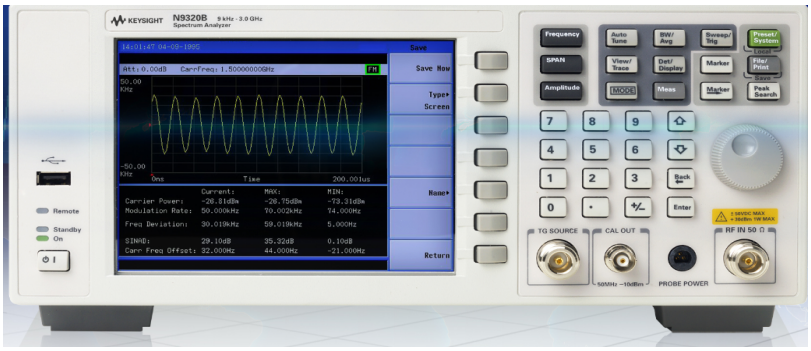
During finding the peak values for chosen frequencies for mini pentatonic scale for frequency 196 Hz, the peak values received in Serial Monitor in Arduino IDE differentiate from each other significantly.

Future work

Spectrum Analyser

“A traditional spectrum analyser searches for signals within a spectral bandwidth and provides snapshots of the signal in the frequency or modulation domain” (www.tek.com. (n.d.). *Spectrum Analyzers & Signal Frequency Analyzers | Tektronix*. [online] Available at: https://www.tek.com/en/products/spectrum-analyzers)

In the presented project, spectrum analyzer can be a handful tool, as it allows to visualize the signal in the frequency or modulation domain. On the market there are many types of spectrum analyzers. Below, there is an example of handheld spectrum analyzer.



Engineerlive.com. (2023). Available at: https://www.engineerlive.com/sites/engineerlive/files/Keysight-N9320B-Spectrum-Analyser.jpg [Accessed 25 Apr. 2023].

Lighting effects

Sonically augmented fountain can be enriched of lighting effect for example by adding DMX lights. (www.youtube.com. (n.d.). *Control DMX Lights with Arduino*. [online] Available at: https://www.youtube.com/watch?v=pdFtoTRiQ-M [Accessed 25 Apr. 2023]s

Magic fountain in Barcelona, Barcelona website. (2017). *Choreographies for the Magic Fountain*. [online] Available at: https://www.barcelona.cat/en/what-to-do-in-bcn/magic-fountain/choreographies-magic-fountain [Accessed 20 Apr. 2023].

Magic fountain in Barcelona, Barcelona website. (2017). *Choreographies for the Magic Fountain*. [online] Available at: https://www.barcelona.cat/en/what-to-do-in-bcn/magic-fountain/choreographies-magic-fountain [Accessed 20 Apr. 2023].

Synthedit – special effects

Synthedit has a variety of effects available such as: Clipper, Delay2, Echo, Flanger, Ring modulator, Soft distortion DH and 4 types of Reverb (DH, JC, SP and Zita). It is worth

playing with some of them.

BIBLIOGRAPHY

Mays, L.W.; Koutsoyiannis, D.; Angelakis, A.N. *A brief history of urban water supply in antiquity*. Water Sci. Technol. Water Supply 2007, 7, 1–12.

Hynynen, A.J.; Juuti, P.S.; Katko, T.S. *Water Fountains in the World scape;* Hynynen, A.J., Juuti, P.S., Katko, T.S., Eds.; International Water History Association (IWHA): Delft, The Netherlands, 2012.

OTB. (n.d.). *Musical fountains*. [online] Available at: https://otb.es/en/musical-fountains/ [Accessed 3 Feb. 2023].

Wikipedia. (2022). *Musical fountain*. [online] Available at: https://en.wikipedia.org/wiki/Musical_fountain [Accessed 3 Feb. 2023].

www.youtube.com. (n.d.). *Bellagio Fountain - Titanic song by Celine Dion*. [online] Available at: https://www.youtube.com/watch?v=YwDvmEfC7OA [Accessed 3 Feb. 2023].

Visconti, P., Costantini, P. and Cavalera, G. (2016). SMART ELECTRONIC SYSTEM FOR DANCING FOUNTAINS CONTROL CAPABLE TO CREATE WATER AND LIGHTING SCENARIOS SYNCHRONIZED WITH A MUSIC TRACK. [online] 11(9). Available at: http://www.arpnjournals.org/jeas/research_papers/rp_2016/jeas_0516_4166.pdf [Accessed 3 Feb. 2023].

Van Renterghem, T., Vanhecke, K., Filipan, K., Sun, K., De Pessemier, T., De Coensel, B., Joseph, W. and Botteldooren, D. (2020). Interactive soundscape augmentation by natural sounds in a noise polluted urban park. *Landscape and Urban Planning*, 194, p.103705. doi:10.1016/j.landurbplan.2019.103705.

www.sciencedirect.com. (n.d.). *Landscape and Urban Planning | Journal | ScienceDirect.com by Elsevier*. [online] Available at: https://www.sciencedirect.com/journal/landscape-and-urban-planning

Xu, C. and Cheng, F. (2022). Virtual Reality Computer-Aided Design of Music Fountain based on Music Feature Recognition. *Computer-Aided Design and Applications*, 20(S1), pp.162–173. doi:10.14733/cadaps.2023.s1.162-173.)

Abboud, R.; Tekli, J.: Integration of nonparametric fuzzy classification with an evolutionary developmental framework to perform music sentiment-based analysis and composition, Soft Computing, 24(13), 2020, 9875-9925

Yoo, M.-J. and Lee, I.-K. (2009). Creating Musical-Fountain Shows. *IEEE Computer Graphics and Applications*, 29(5), pp.6–13. doi:10.1109/mcg.2009.91.

Egorov, A., Mudu, P., Braubach, M., & Martuzzi, M. (2016*). Urban green space and health – a review of evidence*. Copenhagen: WHO Regional Office for Europe.

Fritschi, L., Brown, L., Kim, R., Schwela, D., Kephalopoulos, S. (2011). *Burden of Disease from Environmental Noise—Quantification of Healthy Life Years Lost in Europe*. WHO regional office for Europe

(Partridge, H. (2015). *10 buildings with extraordinary acoustics*. [online] The Spaces. Available at: https://thespaces.com/10-buildings-with-extraordinary-acoustics/.)

SWI swissinfo.ch. (n.d.). *Expo.02: Sound Tower on arteplage Biel-Bienne*. [online] Available at: https://www.swissinfo.ch/eng/expo-02--sound-tower-on-arteplage-biel-bienne/2699686 [Accessed 3 Feb. 2023].

Parchment. (n.d.). *Parchment*. [online] Available at: https://dmsasparchment.com [Accessed 3 Feb. 2023].

The Spaces. (n.d.). *The Spaces*. [online] Available at: https://thespaces.com.

Weisstein, E.W. (n.d.). *Fast Fourier Transform*. [online] mathworld.wolfram.com. Available at: https://mathworld.wolfram.com/FastFourierTransform.html.

www.nti-audio.com. (n.d.). *NTi Audio Solutions for Audio & Acoustics*. [online] Available at: https://www.nti-audio.com/en/.

AZ-Delivery. (n.d.). *Max9814 microphone*. [online] Available at: https://www.az-delivery.de/en/products/max9814-mikrofon [Accessed 31 Jan. 2023].

TutorialsPoint. (n.d.). *Fast Fourier Transform (FFT) on Arduino*. [online] Available at: https://www.tutorialspoint.com/fast-fourier-transform-fft-on-arduino.

Trip101. (2019). *10 Best Fountains In London, England*. [online] Available at: https://trip101.com/article/fountains-in-london.

Michigan Technological University (2019). *Frequencies of Musical Notes*. [online] Mtu.edu. Available at: https://pages.mtu.edu/~suits/notefreqs.html

www.arduinolibraries.info. (n.d.). *arduinoFFT*. [online] Available at: https://www.arduinolibraries.info/libraries/arduino-fft [Accessed 27 Mar. 2023]

(Stoffregen, P. (2023). *PaulStoffregen/SoftwareSerial*. [online] GitHub. Available at: https://github.com/PaulStoffregen/SoftwareSerial/blob/master/SoftwareSerial.h)

(Christensen, P.J. (2023). *ZITA-REV1*) and (www.youtube.com. (n.d.). *SynthEdit Tutorial 16 - Adding a Delay Effect*. [online] Available at: https://www.youtube.com/watch?v=YNhS_ftRlOk)

BAMBOO WATER FOUNTAIN | Relax & Get Your Zen On | White Noise | Tinnitus Relief. (2014). *YouTube*. Available at: https://www.youtube.com/watch?v=aJaZc4E8Y4U [Accessed 2 Mar. 2020].

www.youtube.com. (n.d.). *VID 20230412 134134*. [online] Available at: https://www.youtube.com/watch?v=C2fHkNSJhOs.

www.youtube.com. (n.d.). *VID 20230412 134400*. [online] Available at: https://www.youtube.com/shorts/tqo_rKVEjPg [Accessed 13 Apr. 2023].

(www.tek.com. (n.d.). *Spectrum Analyzers & Signal Frequency Analyzers | Tektronix*. [online] Available at: https://www.tek.com/en/products/spectrum-analyzers)

(2017). *Choreographies for the Magic Fountain*. [online] Available at: https://www.barcelona.cat/en/what-to-do-in-bcn/magic-fountain/choreographies-magic-fountain [Accessed 20 Apr. 2023].

(EDUCBA. (2021). *Matlab fft() | Guide to How Matlab fft() works with Examples*. [online] Available at: https://www.educba.com/matlab-fft/

ccrma.stanford.edu. (n.d.). *Essentials of the MIDI protocol*. [online] Available at: https://ccrma.stanford.edu/~craig/articles/linuxmidi/misc/essenmidi.html [Accessed 19 Apr. 2023].

Anon, (n.d.). Available at:

https://www.researchgate.net/publication/268376151_WEB_BASED_INTELLIGENT_APPOINTMENT_SYSTEM

(www.twi-global.com. (n.d.). *What is Prototyping? (A Complete Guide)*. [online] Available at: https://www.twi-global.com/technical-knowledge/faqs/what-is-prototyping

" Anon, (n.d.). *What Is A Prototype Model? (Methodology, Types And Uses)*. [online] Available at: https://in.indeed.com/career-advice/career-development/what-is-prototype-model [Accessed 23 Apr. 19AD].

APPENDIX A

```
#include "arduinoFFT.h"

#define SAMPLES 128             //Must be a power of 2

#define SAMPLING_FREQUENCY 1000 //Hz, must be less than 10000 due to ADC

arduinoFFT FFT = arduinoFFT();

unsigned int sampling_period_us;

unsigned long microseconds;

double vReal[SAMPLES];

double vImag[SAMPLES];

void setup() {

 Serial.begin(115200);

 sampling_period_us = round(1000000*(1.0/SAMPLING_FREQUENCY));

}

void loop() {
```

```
/*SAMPLING*/

for(int i=0; i<SAMPLES; i++)

{

microseconds = micros();    //Overflows after around 70 minutes!

vReal[i] = analogRead(0);

vImag[i] = 0;

while(micros() < (microseconds + sampling_period_us)){

 }

}

/*FFT*/

FFT.Windowing(vReal, SAMPLES, FFT_WIN_TYP_HAMMING, FFT_FORWARD);

FFT.Compute(vReal, vImag, SAMPLES, FFT_FORWARD);

FFT.ComplexToMagnitude(vReal, vImag, SAMPLES);

double peak = FFT.MajorPeak(vReal, SAMPLES, SAMPLING_FREQUENCY);

for(int i=0; i<(SAMPLES/2); i++)

{

 /*View all these three lines in serial terminal to see which frequencies has which amplitudes*/

      Serial.print((i * 1.0 * SAMPLING_FREQUENCY) / SAMPLES, 1);

      Serial.print(" ");

      Serial.println(vReal[i], 1);    //View only this line in serial plotter to visualize the bins
```

```
 }

 delay(500);  //Repeat the process every second OR:

 //while(1);       //Run code once

}
```

APPENDIX B

```
#include "arduinoFFT.h"

#define SAMPLES 128             //Must be a power of 2

#define SAMPLING_FREQUENCY 1000 //Hz, must be less than 10000 due to ADC

arduinoFFT FFT = arduinoFFT();

unsigned int sampling_period_us;

unsigned long microseconds;

double vReal[SAMPLES];

double vImag[SAMPLES];

void setup() {

 Serial.begin(115200);

 sampling_period_us = round(1000000*(1.0/SAMPLING_FREQUENCY));

}

void loop() {

 /*SAMPLING*/

 for(int i=0; i<SAMPLES; i++)
```

```
{

microseconds = micros();    //Overflows after around 70 minutes!

vReal[i] = analogRead(0);

vImag[i] = 0;

while(micros() < (microseconds + sampling_period_us)){

 }

}

/*FFT*/

FFT.Windowing(vReal, SAMPLES, FFT_WIN_TYP_HAMMING, FFT_FORWARD);

FFT.Compute(vReal, vImag, SAMPLES, FFT_FORWARD);

FFT.ComplexToMagnitude(vReal, vImag, SAMPLES);

double peak = FFT.MajorPeak(vReal, SAMPLES, SAMPLING_FREQUENCY);

/*PRINT RESULTS*/
Serial.print("peak = "),
Serial.println(peak);     //Print out what frequency is the most dominant.

for(int i=0; i<(SAMPLES/2); i++)

{

 /*View all these three lines in serial terminal to see which frequencies has which amplitudes*/

      Serial.print((i * 1.0 * SAMPLING_FREQUENCY) / SAMPLES, 1);

      Serial.print(" ");

      Serial.println(vReal[i], 1);    //View only this line in serial plotter to visualize the bins
```

```
}

 delay(500);  //Repeat the process every second OR:

 //while(1);       //Run code once

}
```

APPENDIX C

```
#include "arduinoFFT.h"

#include <SoftwareSerial.h>

SoftwareSerial mySerial(2, 3); // RX, TX

#define SAMPLES 128 //Must be a power of 2

#define SAMPLING_FREQUENCY 1000 //Hz, must be less than 10000 due to ADC

arduinoFFT FFT = arduinoFFT();

unsigned int sampling_period_us;

unsigned long microseconds;

double vReal[SAMPLES];

double vImag[SAMPLES];

void setup() {

Serial.begin(115200);

mySerial.begin(31250);////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////

randomSeed(analogRead(5));

sampling_period_us = round(1000000*(1.0/SAMPLING_FREQUENCY));
```

```
}

void loop() {

/*SAMPLING*/

int ran =  random(30,700);

for(int i=0; i<SAMPLES; i++)

{

microseconds = micros(); //Overflows after around 70 minutes!

vReal[i] = analogRead(0);

vImag[i] = 0;

while(micros() < (microseconds + sampling_period_us)){

}

}

/*FFT*/

FFT.Windowing(vReal, SAMPLES, FFT_WIN_TYP_HAMMING, FFT_FORWARD);

FFT.Compute(vReal, vImag, SAMPLES, FFT_FORWARD);

FFT.ComplexToMagnitude(vReal, vImag, SAMPLES);

double peak = FFT.MajorPeak(vReal, SAMPLES, SAMPLING_FREQUENCY);

/*PRINT RESULTS*/

if (peak == 222.60 ) {  // A3

noteOn(0x90, 57, 0x45);
```

```
delay(10);

noteOn(0x90,57, 0x00);
}

if (peak == 265.89) {  // C4

noteOn(0x90, 60, 0x45);

delay(1000);

noteOn(0x90,60, 0x00);
}

if (peak == 298.37) {  // D4

noteOn(0x90, 62, 0x45);

delay(10);

noteOn(0x90,62, 0x00);
}

if (peak == 334.35) {  // E4

noteOn(0x90, 64, 0x45);

delay(10);

noteOn(0x90,64, 0x00);
}

if (peak == 397.82) { // G4

noteOn(0x90,67, 0x45);

delay(10);

noteOn(0x90,67, 0x00);
```

```
}

if (peak == 447.11) { // A4

noteOn(0x90,69, 0x45);

delay(10);

noteOn(0x90,69, 0x00);
}

if (peak == 472.91 ) { // A#4

noteOn(0x90,70, 0x45);

delay(10);

noteOn(0x90,70, 0x00);
}

Serial.print("peak = ");

Serial.println(peak); //Print out what frequency is the most dominant.

for(int i=0; i<(SAMPLES/2); i++)

{

/*View all these three lines in serial terminal to see which frequencies has which amplitudes*/

Serial.print((i * 1.0 * SAMPLING_FREQUENCY) / SAMPLES, 1);

Serial.print(" ");

Serial.println(vReal[i], 1); //View only this line in serial plotter to visualize the bins

}

delay (ran); //Repeat the process every second OR:
```

```
//while(1); //Run code once=

}

// plays a MIDI note. Doesn't check to see that cmd is greater than 127, or that

// data values are less than 127:

void noteOn(int cmd, int pitch, int velocity) {

mySerial.write(cmd);

mySerial.write(pitch);

mySerial.write(velocity);

}
```

APPENDIX D

```
#include "arduinoFFT.h"
#include "SoftwareSerialTX.h"

SoftwareSerialTX mySerial(2); // RX, TX

#define SAMPLES 128 //Must be a power of 2

#define SAMPLING_FREQUENCY 1000 //Hz, must be less than 10000 due to ADC

arduinoFFT FFT = arduinoFFT();

unsigned int sampling_period_us;

unsigned long microseconds;

double vReal[SAMPLES];

double vImag[SAMPLES];

void setup() {
```

```
Serial.begin(115200);

mySerial.begin(31250);////////////////////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////

randomSeed(analogRead(5));

sampling_period_us = round(1000000*(1.0/SAMPLING_FREQUENCY));

}

void loop() {

/*SAMPLING*/

int ran =  random(30,200);

for(int i=0; i<SAMPLES; i++)

{

microseconds = micros(); //Overflows after around 70 minutes!

vReal[i] = analogRead(0);

vImag[i] = 0;

while(micros() < (microseconds + sampling_period_us)){

}

}

/*FFT*/

FFT.Windowing(vReal, SAMPLES, FFT_WIN_TYP_HAMMING, FFT_FORWARD);

FFT.Compute(vReal, vImag, SAMPLES, FFT_FORWARD);
```

```
FFT.ComplexToMagnitude(vReal, vImag, SAMPLES);

double peak = FFT.MajorPeak(vReal, SAMPLES, SAMPLING_FREQUENCY);

Serial.println("Just Before");

/*PRINT RESULTS*/

if (peak > 221 && peak < 225) {  // A3

Serial.println("Hello");

noteOn(0x90, 57, 0x45);

delay(10);

noteOn(0x90,57, 0x00);
}


if (peak > 263 && peak < 267) {  // C4

noteOn(0x90, 60, 0x45);

delay(10);

noteOn(0x90,60, 0x00);
}

if (peak > 296 && peak < 300) {  // D4

noteOn(0x90,62,0x45);

delay(10);

noteOn(0x90,62,0x00);
}

if (peak > 333 && peak < 337) {  // E4
```

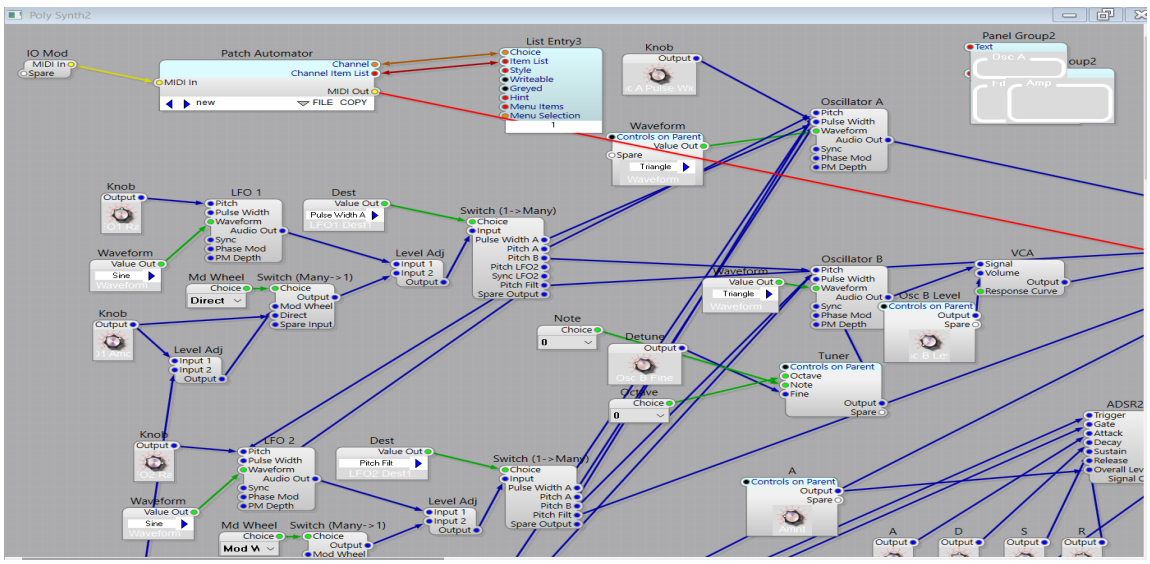
```
noteOn(0x90, 64, 0x45);

delay(10);

noteOn(0x90,64, 0x00);
}


// if (peak > 391 && peak > 397) { // G4

// noteOn(0x90,67, 0x45);

// delay(10);

// noteOn(0x90,67, 0x00);
// }


if (peak > 445 && peak < 449) { // A4

noteOn(0x90,69, 0x45);

delay(10);

noteOn(0x90,69, 0x00);
}


if (peak > 470 && peak < 474) { // A#4

noteOn(0x90,70, 0x45);

delay(10);

noteOn(0x90,70, 0x00);
}
//////////////////
if (peak > 110.5  && peak < 112.5) {  // half  A3

Serial.println("Hello");
```

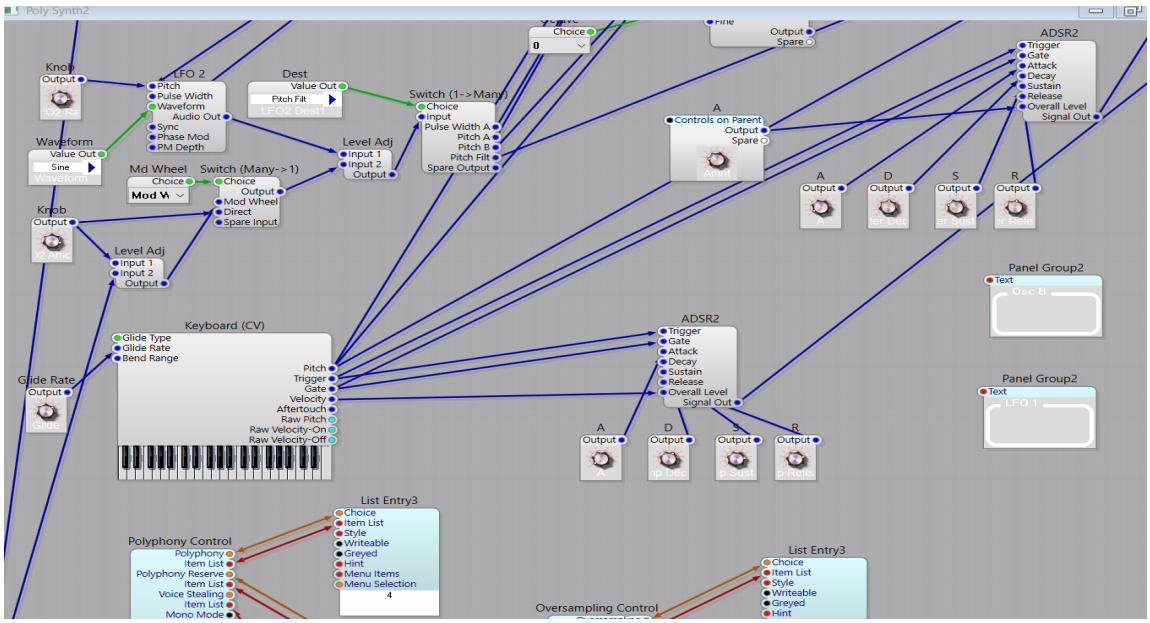
```
noteOn(0x90, 57, 0x45);

delay(10);

noteOn(0x90,57, 0x00);
}

if (peak > 131.5 && peak < 133.5 ) {  //  half C4

noteOn(0x90, 60, 0x45);

delay(10);

noteOn(0x90,60, 0x00);
}

if (peak > 148 && peak < 150) {  // half D4

noteOn(0x90,62,0x45);

delay(10);

noteOn(0x90,62,0x00);
}

if (peak > 166.5 && peak < 168.5) {  // half  E4

noteOn(0x90, 64, 0x45);

delay(10);

noteOn(0x90,64, 0x00);
}

// if (peak > 195  && peak > 197) { //half  G4

// noteOn(0x90,67, 0x45);

// delay(10);
```

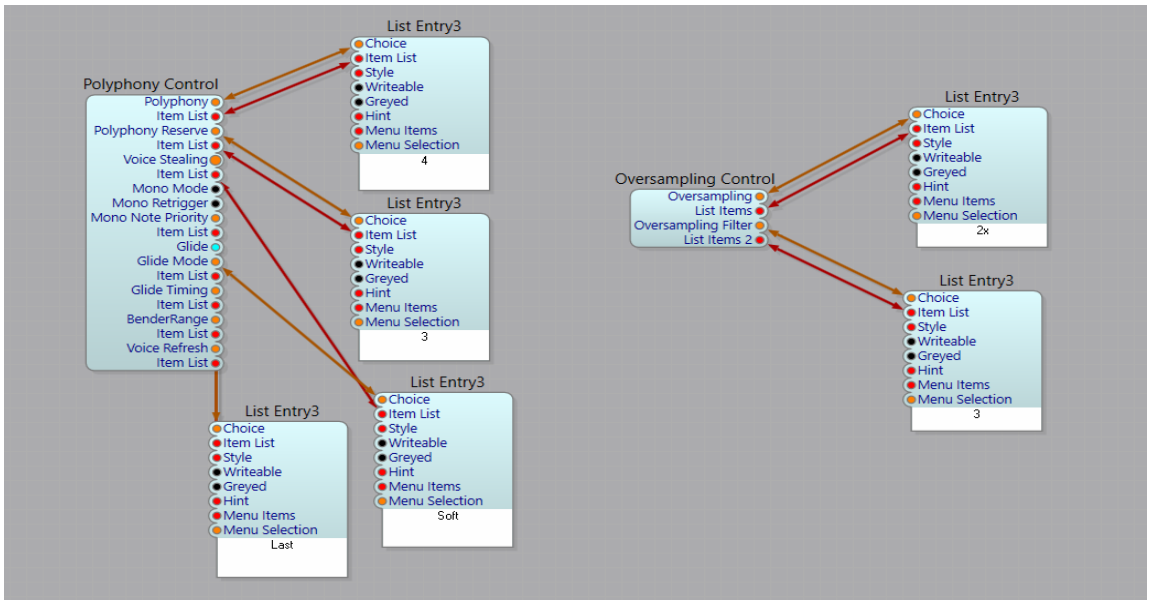
```
// noteOn(0x90,67, 0x00);
// }

if (peak > 222.5 && peak < 224.5) { // half A4

noteOn(0x90,69, 0x45);

delay(10);

noteOn(0x90,69, 0x00);
}

if (peak > 235 && peak < 237.25) { // half A#4

noteOn(0x90,70, 0x45);

delay(10);

noteOn(0x90,70, 0x00);
}


Serial.print("peak = ");

Serial.println(peak); //Print out what frequency is the most dominant.

for(int i=0; i<(SAMPLES/2); i++)

{

/*View all these three lines in serial terminal to see which frequencies has which amplitudes*/

//Serial.print((i * 1.0 * SAMPLING_FREQUENCY) / SAMPLES, 1);

//Serial.print(" ");

//Serial.println(vReal[i], 1); //View only this line in serial plotter to visualize the bins
```

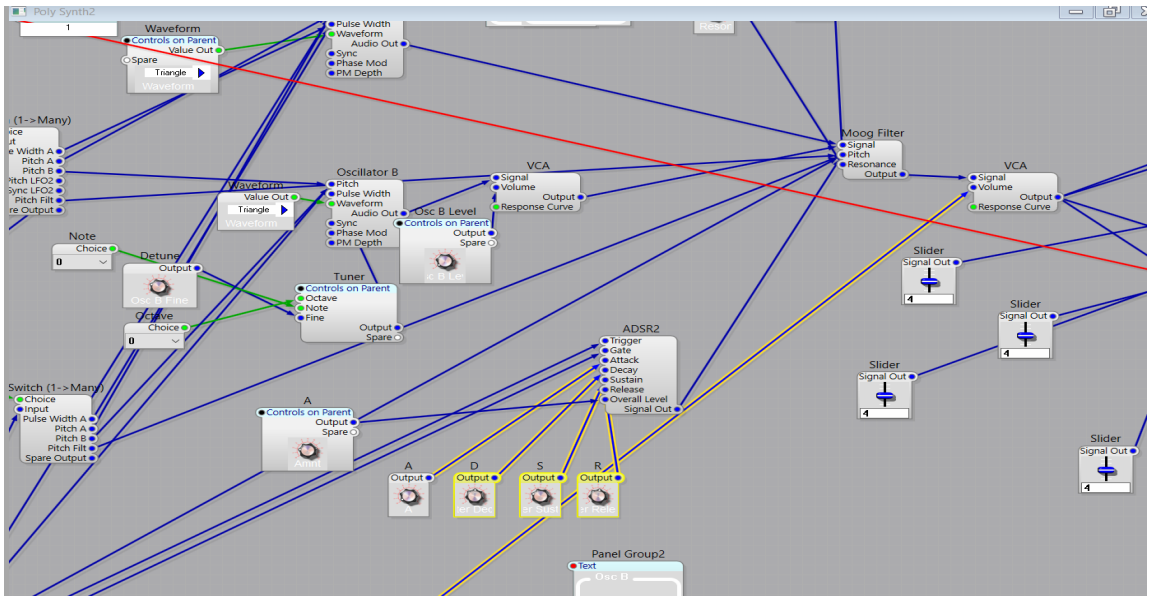
```
}

delay (ran); //Repeat the process every second OR:

//while(1); //Run code once=

}

// plays a MIDI note. Doesn't check to see that cmd is greater than 127, or that

// data values are less than 127:

void noteOn(int cmd, int pitch, int velocity) {

mySerial.write(cmd);

mySerial.write(pitch);

mySerial.write(velocity);

}
```

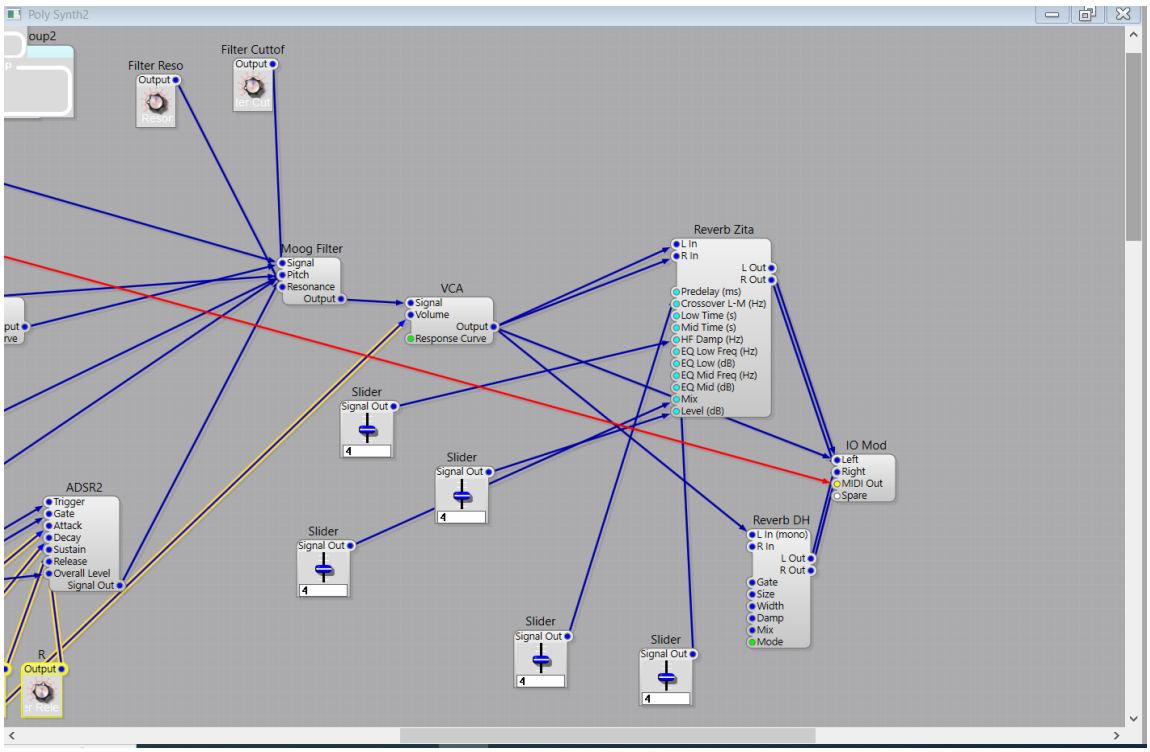
APPENDIX E




Poly Synth2
Knob
LFO 2
Dest
Switch (1->Many)
ADSR2
Waveform
Level Adj
Md Wheel
Switch (Many->1)
Keyboard (CV)
Glide Rate
Panel Group2
Polyphony Control
List Entry3
Oversampling Control


Polyphony Control
List Entry3
Oversampling Control
Soft
Last
2x
3
4


Poly Synth2
Waveform
Controls on Parent
Value Out
Spare
Triangle
Waveform
Pulse Width
Waveform
Audio Out
Sync
Phase Mod
PM Depth
(1->Many)
Pitch A
Pitch B
Pitch Filt
Oscillator B
Pitch
Pulse Width
Waveform
Audio Out
Sync
Phase Mod
PM Depth
VCA
Signal
Volume
Output
Response Curve
Osc B Level
Controls on Parent
Output
Spare
Moog Filter
Signal
Pitch
Resonance
Output
Note
Choice
Detune
Octave
Tuner
Octave
Note
Fine
Slider
Signal Out
ADSR2
Trigger
Gate
Attack
Decay
Sustain
Release
Overall Level
Signal Out
Switch (1->Many)
Input
A
D
S
R
Panel Group2
Text


Poly Synth2
Filter Reso
Filter Cuttof
Output
Moog Filter
Signal
Pitch
Resonance
Output
VCA
Signal
Volume
Output
Response Curve
Reverb Zita
L In
R In
L Out
R Out
Predelay (ms)
Crossover L-M (Hz)
Low Time (s)
Mid Time (s)
HF Damp (Hz)
EQ Low Freq (Hz)
EQ Low (dB)
EQ Mid Freq (Hz)
EQ Mid (dB)
Mix
Level (dB)
IO Mod
Left
Right
MIDI Out
Spare
Slider
Signal Out
ADSR2
Trigger
Gate
Attack
Decay
Sustain
Release
Overall Level
Signal Out
Reverb DH
L In (mono)
R In
L Out
R Out
Gate
Size
Width
Damp
Mix
Mode
R
Output