Monophonic Guitar Synthesizer via Mobile App

Edgar García Leyva[1]
Instituto Politécnico Nacional. SEPI-ESCOM
Mexico City, Mexico

Elena Fabiola Ruiz Ledesma[2]
Instituto Politécnico Nacional.
ESCOM, UPIICSA
Mexico City, Mexico

Rosaura Palma Orozco[3]
Lorena Chavarría Báez[4]
Instituto Politécnico Nacional.
ESCOM, Mexico City, Mexico

***Abstract*—In the guild of guitarists, it is common to work with guitar synthesizers because the emulation of a great variety of sounds that are produced by different musical instruments, starting from just playing the guitar, which means, a piece of music is played with a guitar, but other musical instruments are actually heard such as, a saxophone, a violin, a piano or percussions, depending on the instrument that has been selected. The problem that arises in this article is that synthesizers are expensive and due to their size, the transportation of the equipment is often impractical. As mentioned, the development of a mobile application that has the function of a monophonic synthesizer is proposed as a solution. In this way, the cost is greatly reduced, and additionally, the user is able to install the application on a mobile device with Android operating system and connect it to an electric or electro-acoustic guitar through an audio interface; obtaining as a result, a functional technological instrument by offering guitarists an alternative with respect to conventional synthesizers. The construction of this application used the Fast Fourier Transform Radix-2 as a signal recognition algorithm, which allowed obtaining the fundamental frequencies generated by the guitar, which were transformed into MIDI notation and later used in sound emulation.**

Keywords*—Monophonic synthesizer; guitar; sound emulation; mobile application

I. Introduction

The advent of tablets and smart cell phones has opened a range of possibilities in all areas of knowledge. In the field of music, technological development has occupied a privileged place by having mobile applications that help in tuning musical instruments, in measuring time to practice, in recording and editing music, among other aspects.

The rise of technology has allowed the creation of tools in order to support musicians in a vast diversity of ways. The tool that is interesting to highlight in this article is related to the emulation of monophonic sounds of musical instruments or other types of sounds, which is obtained through the use of synthesizers. The Royal Academy of the Spanish Language defines synthesizer as: "Electronic musical instrument capable of producing sounds of any frequency and intensity and combining them with harmonics, thus providing sounds of any known instrument, or sound effects that do not correspond to any conventional instrument" [1]. Synthesizers are very useful devices because when connected to the guitar, are able to provide a great variety of sounds, increasing the possibilities of musical interpretation of the guitarist.

Two of the problems that arise are its portability and its high cost; Due to the mentioned, based on computer science and mathematics, it was decided to develop a monophonic synthesizer, using a mobile application, which allows the guitarist to use a synthesizer through a mobile device such as a smart cell phone or a tablet, which can be transported easily, without having to make an additional expense.

Music can be classified into two main categories: monophonic and polyphonic. Monophonic music is made up of a single melodic line, which means, only one musical note sounds at a certain time, while polyphonic music is made up of more than one melodic line, which means, two or more musical notes sound at the same time [2]. The present study focuses on monophonic music.

The guitar is a musical instrument that allows its player to express musical notes in different ways. The guitar, like other instruments such as the piano or the violin, generates analog sounds, which need to be converted to digital sounds in order to be read by the computer. From this digitization, it is possible to apply signal recognition techniques for different purposes and in this case, to detect the fundamental frequencies generated by the guitar. A transformation was applied to these frequencies using the notation mentioned in the Musical Instrument Digital Interface (MIDI), to obtain a discrete representation of the musical notes, which are defined in an interval that goes from 0 to 127, where each number corresponds to a musical note.

The overall objective of this study was to develop a portable tool that allows the guitarist to emulate monophonic sounds of other musical instruments or other types of sounds using the guitar, all this through computational and mathematical techniques. For this purpose, the following specific objectives were proposed:

- Recognize frequencies generated by guitars making use of hardware and software resources of mobile devices.
- Transform frequencies generated by guitars to MIDI notation.
- Emulate monophonic sounds of musical instruments or other sounds based on the obtained MIDI notes.

This article is divided into 4 sections. The second section shows some synthesizers available on the market, as well as some application programming interfaces that have been developed in order to recognize signals and to support the execution of MIDI sounds. The third section deals with the theoretical references about the recognition of monophonic sounds and the transformation of frequency to MIDI notation. The fourth section shows the methodology used for the

development of the monophonic synthesizer following the stages of the incremental Software Engineering model. Subsequently, the results obtained from the tests carried out are shown and finalized with the conclusions.

II. Related Work

At present, there are independent synthesizers for guitar, which means, mounted on electronic circuits, some of them are: MEL9 [3], SY-300 [4], GR-55 [5], among others; However, when it comes to mobile applications that perform the function of a guitar synthesizer on the Android operating system, there are no formal alternatives to it.

On the other hand, there are some application programming interfaces (APIs) aimed at the Android operating system, which can be useful for the construction of a synthesizer, for example, those that serve to perform frequency recognition and those that serve to execute MIDI sounds, some of these interfaces are: TarsosDSP [6] and MIDI Driver [7] respectively.

In the present mobile application, it was chosen the frequency recognition through the Fast Fourier Transform Radix-2 algorithm, according to [8], making use of the native Java development kit, while for the execution of MIDI sounds, the model that was done in [7] was retaken.

III. Theoretical Aspects

In this section, reference is made to the techniques used to recognize monophonic sounds and, on the other hand, the digital interface of musical instruments (MIDI) is presented, which shows a notation that serves to discretize frequencies.

A. Monophonic Sound Recognition

Due to in this article the monophonic sounds generated by the guitar are taken as a basis, some algorithms that can be used for the recognition of monophonic sounds are specifically mentioned.

There are mainly two approaches that are used to perform monophonic sound recognition. One of them consists in analyzing the signal samples in the time domain, and the other in analyzing them in the frequency domain. A widely used method in the time domain is autocorrelation, which compares a signal with delayed versions of itself at successive intervals to find the highest amplitudes within the signal and measure the distances between them. Through these distances the period of the wave can be inferred, and with it, the present frequency of the monophonic sound can be detected. On the other hand, there is the analysis of signals in the frequency domain, where algorithmic implementations of the Discrete Fourier Transform are used, with which a set of frequency intervals and their amplitudes are obtained. A simple way to detect the frequency of the monophonic sound present quickly is to select the frequency with the greatest amplitude [9].

In this article, the signals in the frequency domain are analyzed, so some algorithms that can be used to recognize the monophonic sounds generated by the guitar in that domain are specifically mentioned.

B. Fast Fourier Transform (FFT)

Fast Fourier Transform is an efficient mathematical implementation of the Discrete Fourier Transform (DFT), which is a particular case of the Fourier Transform for sequences of finite length in which the spectrum is evaluated only in a few specific frequencies, and therefore, a discrete spectrum is obtained [10, 11].

Over time, several FFT algorithms have been developed such as: prime factor, split radix, vector radix, split vector radix, Winograd Fourier transform, etc. [12].

The FFT algorithm used to develop the mobile application of the monophonic synthesizer is the Radix-2. The Radix-2 algorithm is considered the most used for the FFT calculation, it works when the number of data samples is a power of 2, in case the number of samples does not satisfy this criterion, the missing spaces are filled with value 0, this does not alter the calculated frequency spectrum. The input and output of an FFT are expressed in complex numbers, in this case, for its implementation, two arrangements are accepted to store the real and imaginary components in them, when using the recording tools of mobile devices, all the bytes of audio information are obtained, which are used within the real components, while the imaginary components are always filled with zeros, the output of the algorithm is contained in two other arrays, one corresponding to each type of component, where the frequency spectra are stored, since only the actual samples are used for input, only the first half of the components need to be analyzed. Each component of the frequency spectrum is related to the previous component, since it is the sampling frequency divided by the number of samples of the FFT [8].

In order to recognize monophonic sounds, the power or amplitude spectrum (Xp) is analyzed, selecting the frequency f with the greatest amplitude, which is calculated through the sum of the squares of its real (Xreal) and imaginary (Ximag) components, as shown in (1),

$$Xp = Xreal(f)^2 + Ximag(f)^2 \quad (1)$$

The FFT algorithm used to develop the mobile application is Radix-2, which means that it must work on a group of samples whose number is a power of two [8].

C. MIDI

Musical Instrument Digital Interface (MIDI) is a music notation system that allows computers to communicate with musical synthesizers. MIDI files contain instructions to create the pitch, volume and duration of the notes, this based on a sequence of events called: note_on and note_off [13].

Musical notes are not encoded by their names, instead numbers from 0 to 127 are assigned as shown in Table I. For example, the number 57 corresponds to a musical note A with a frequency of 220 Hertz (Hz).

Equation (2) shows the transformation of frequency in Hz to MIDI note,

$$MIDINote = round(69 + 12 \times \log_2 (f / 440)) \quad (2)$$

TABLE I. MIDI NOTES ASSOCIATED WITH ITS NAME AND FREQUENCY IN HZ [15]

Name	MIDI note	Frequency (Hz)
D	38	73.42
D♯/E ♭	39	77.78
E	40	82.41
F	41	87.31
F♯/G ♭	42	92.50
G	43	98.00
G♯/A ♭	44	103.83
A	45	110.00
A♯/B ♭	46	116.54
B	47	123.47
C	48	130.81
C♯/D ♭	49	138.59
D	50	146.83
D♯/E ♭	51	155.56
E	52	164.81
F	53	174.61
F♯/G ♭	54	185.00
G	55	196.00
G♯/A ♭	56	207.65
A	57	220.00
A♯/B ♭	58	233.08
B	59	246.94
C	60	261.63
C♯/D ♭	61	277.18
D	62	293.66
D♯/E ♭	63	311.13
E	64	329.63
F	65	349.23
F♯/G ♭	66	369.99
G	67	392.00
G♯/A ♭	68	415.30
A	69	440.00
A♯/B ♭	70	466.16
B	71	493.88
C	72	523.25
C♯/D ♭	73	554.37
D	74	587.33
D♯/E ♭	75	622.25
E	76	659.26
F	77	698.46
F♯/G ♭	78	739.99
G	79	783.99
G♯/A ♭	80	830.61
A	81	880.00
A♯/B ♭	82	932.33
B	83	987.77
C	84	1046.50
C♯/D ♭	85	1108.73
D	86	1174.66
D♯/E ♭	87	1244.51
E	88	1318.51
F	89	1396.91
F♯/G ♭	90	1479.98
G	91	1567.98
G♯/A ♭	92	1661.22
A	93	1760.00

Where round is a function of rounding to one digit, factor 12 is the resulting linear pitch space per octave, factor 69 is note A (440 Hz), which is taken as reference, $\log_2$ is used according to the logarithmic pitch perception in humans and the variable f is the input frequency that will be converted to a MIDI note [14].

General MIDI is a standardized specification for electronic musical instruments that respond to MIDI messages. General MIDI was developed by the American MIDI Manufacturers Association (MMA) and the Japan MIDI Standards Committee (JMSC) and first published in 1991 [16].

Within the general MIDI specification 128 sounds of musical instruments or other types of sounds are included, and these are divided in sections such as: Piano, Chromatic Percussion, Organ, Guitar, Bass, Strings, Ensemble, Brass, Reed, Pipe, Synth Lead, Synth Pad, Synth Effects, Ethnic, Percussive and Sound Effects which are used in this mobile application and are shown in Table II.

TABLE II. SOUNDS OF THE GENERAL MIDI SPECIFICATION [16]

00 - Acoustic Grand Piano	32 - Acoustic Bass	64 - Soprano Sax	96 - FX 1 (rain)
01 - Bright Acoustic Piano	33 - Electric Bass (finger)	65 - Alto Sax	97 - FX 2 (soundtrack)
02 - Electric Grand Piano	34 - Electric Bass (pick)	66 - Tenor Sax	98 - FX 3 (crystal)
03 - Honky-tonk Piano	35 - Fretless Bass	67 - Baritone Sax	99 - FX 4 (atmosphere)
04 - Electric Piano 1	36 - Slap Bass 1	68 – Oboe	100 - FX 5 (brightness)
05 - Electric Piano 2	37 - Slap Bass 2	69 - English Horn	101 - FX 6 (goblins)
06 - Harpsichord	38 - Synth Bass 1	70 – Bassoon	102 - FX 7 (echoes)
07 - Clavi	39 - Synth Bass 2	71 – Clarinet	103 - FX 8 (sci-fi)
08 - Celesta	40 - Violin	72 – Piccolo	104 - Sitar
09 - Glockenspiel	41 - Viola	73 – Flute	105 - Banjo
10 - Music Box	42 - Cello	74 – Recorder	106 - Shamisen
11 - Vibraphone	43 - Contrabass	75 - Pan Flute	107 - Koto
12 - Marimba	44 - Tremolo Strings	76 - Blown Bottle	108 - Kalimba
13 - Xylophone	45 - Pizzicato Strings	77 – Shakuhachi	109 - Bag pipe
14 - Tubular Bells	46 - Orchestral Harp	78 – Whistle	110 - Fiddle
15 - Dulcimer	47 - Timpani	79 – Ocarina	111 - Shanai
16 - Drawbar Organ	48 - String Ensemble 1	80 - Lead 1 (square)	112 - Tinkle Bell
17 - Percussive Organ	49 - String Ensemble 2	81 - Lead 2 (sawtooth)	113 - Agogô
18 - Rock Organ	50 - Synth Strings 1	82 - Lead 3 (calliope)	114 - Steel Drums
19 - Church Organ	51 - Synth Strings 2	83 - Lead 4 (chiff)	115 - Woodblock
20 - Reed Organ	52 - Choir Aahs	84 - Lead 5 (charang)	116 - Taiko Drum
21 - Accordion	53 - Voice Oohs	85 - Lead 6 (voice)	117 - Melodic Tom
22 - Harmonica	54 - Synth Voice	86 - Lead 7 (fifths)	118 - Synth Drum
23 - Tango Accordion	55 - Orchestra Hit	87 - Lead 8 (bass + lead)	119 - Reverse Cymbal
24 - Acoustic Guitar (nylon)	56 - Trumpet	88 - Pad 1 (new age)	120 - Guitar Fret Noise
25 - Acoustic Guitar (steel)	57 - Trombone	89 - Pad 2 (warm)	121 - Breath Noise
26 - Electric Guitar (jazz)	58 – Tuba	90 - Pad 3 (polysynth)	122 - Seashore
27 - Electric Guitar (clean)	59 - Muted Trumpet	91 - Pad 4 (choir)	123 - Bird Tweet
28 - Electric Guitar (muted)	60 - French Horn	92 - Pad 5 (bowed)	124 - Telephone Ring
29 - Overdriven Guitar	61 - Brass Section	93 - Pad 6 (metallic)	125 - Helicopter
30 - Distortion Guitar	62 - Synth Brass 1	94 - Pad 7 (halo)	126 - Applause
31 - Guitar harmonics	63 - Synth Brass 2	95 - Pad 8 (sweep)	127 - Gunshot

IV. METHODOLOGY

The mobile application was developed using the phases of the incremental Software Engineering model, which applies linear sequences in a staggered manner as the calendar of activities progresses. Each linear sequence produces deliverable software increments [17]. The diagram in Fig. 1 describes the stages used for this development.

According to what Mall [18] points out, first of all a simple system is built and delivered which implements only a few basic characteristics. During a few successive iterations, improved versions are deployed and delivered, until the desired system is finally realized.

The software requirements are first divided into several modules or features that can be built and delivered incrementally. This is graphically represented in Fig. 2.

Returning to what Pressman and Mall [17, 18] point out, 3 modules were created as part of the development of the mobile application.

In the first module, the recognition of the frequencies generated by the guitar was carried out, for which the Fast Fourier Transform Radix-2 algorithm was applied, making use of the hardware and software resources of mobile devices. When playing a musical note with the guitar, the fundamental frequency corresponding to said musical note was obtained in real time, showing it on the mobile application interface.

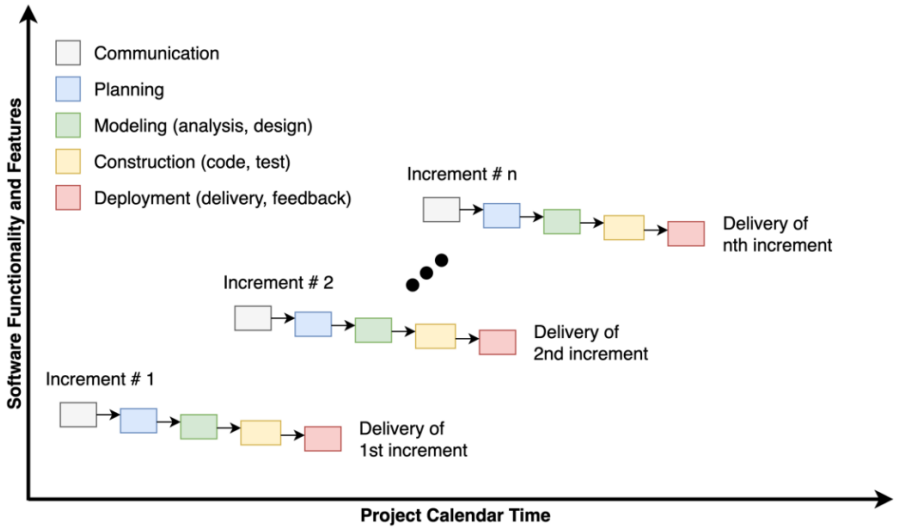

Fig. 1. Diagram of the Incremental Model.

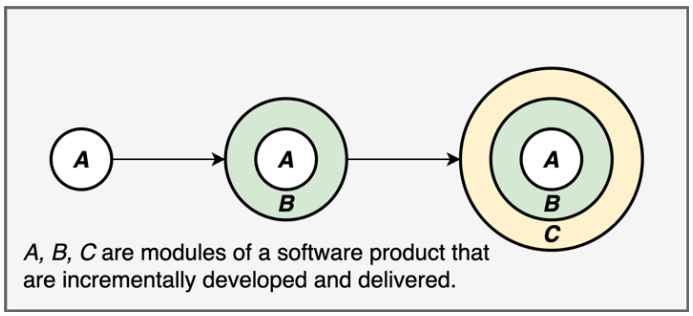

Fig. 2. Incremental Software Development.

In the second module, the transformation of frequencies to MIDI notes was carried out, for which equation 1 shown in the MIDI section was applied, obtaining as a result, discrete values of frequencies within a range from 0 to 127, showing the aforementioned transformation together with at the fundamental frequency in the mobile application interface.

The third module allowed to emulate the monophonic sounds of musical instruments or other types of sounds, based on the MIDI notes obtained. The note_on and note_off events of the digital interface of musical instruments were used to execute and stop the MIDI notes. Consequently, the moment the user plays a musical note with the guitar, the note_on event is activated, with which the sound chosen by the user within the 128 included in the mobile application must be heard; for example, a saxophone, a trumpet, a violin, among others; whereas when there is an absence of sound, the note_off event was activated to keep the mobile application silent.

For the construction of all these modules, the Java programming language was used together with the Android Development Kit (SDK).

A. *Logical Block Diagram of the Structure of the Mobile Application*

Fig. 3 shows the block diagram of the mobile application. As can be seen, an audio input is required, which goes through a sampling process to obtain the discretized signal, making use of the audio recording tools offered by the Android operating system. Subsequently, the filter allows to eliminate the peaks of the signal, which means, it eliminates the unwanted noise, and then to apply the Fast Fourier Transform and thus obtain the fundamental frequencies. From these fundamental frequencies, the transformation to MIDI notation is carried out. From this moment, the sound chosen by the user can be emulated through the execution of the note_on and note_off events.

B. *Operation of the Mobile Application*

Fig. 4 represents the operation of the mobile application. In order to use it, it is necessary to connect it to an electric or electroacoustic guitar through an audio interface, the last one is connected to an audio output device such as an amplifier.

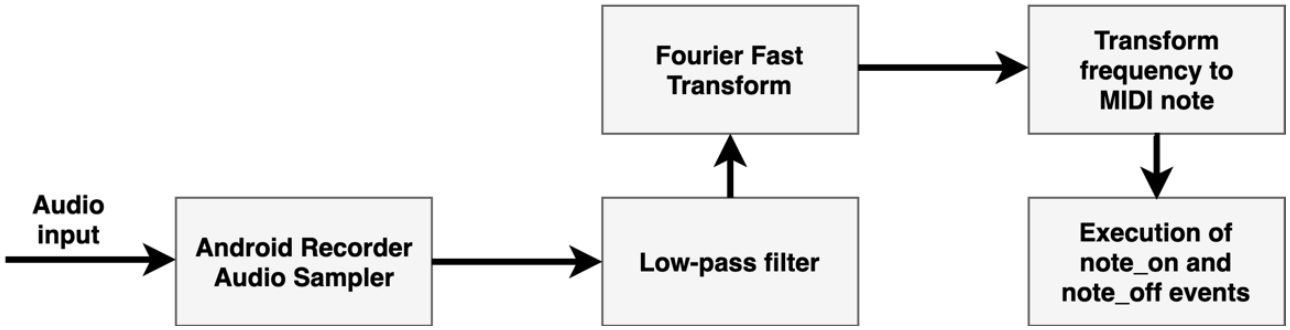

Fig. 3. Block Diagram of the Mobile Application.

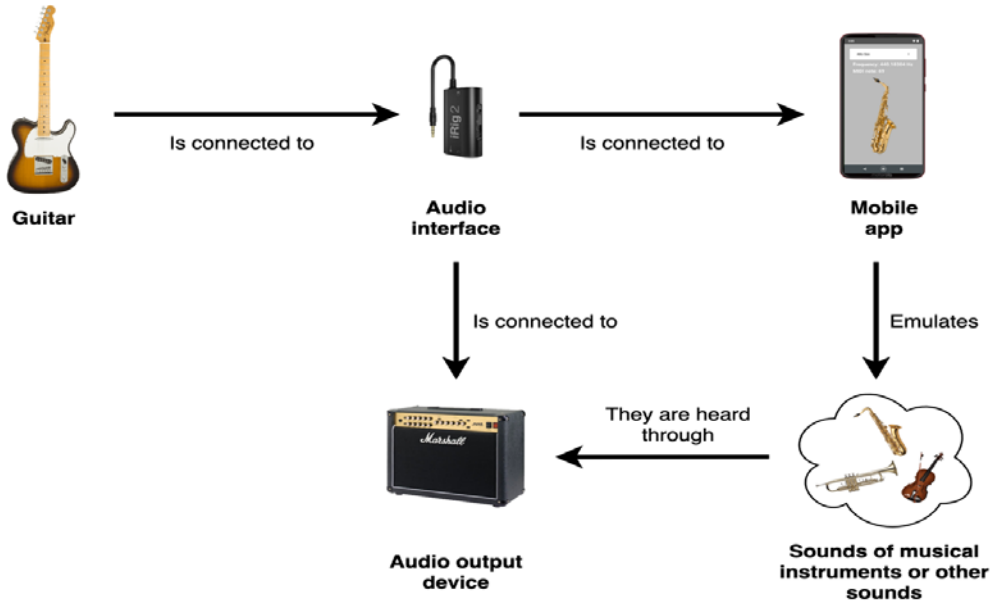

Fig. 4. Mobile Application Operation.

V. Results

The mobile application has been developed using the Java programming language in order to run on mobile devices with the Android operating system. The tests were carried out on a device from the Motorola brand, a Moto G7 Plus model with 64 GB of storage and 4 GB of RAM, with the Android 10 (Android Q) operating system installed.

The work was carried out with a sample of three guitarists to carry out the tests of the operation of the mobile application. Two of the guitarists used an electric guitar and the remaining guitarist used an electro-acoustic guitar. The musical instruments were connected to the mobile application through an iRig 2 audio interface. The guitarists selected in the list of the main interface of the mobile application, several of the 128 sounds that were available to be emulated and they played the guitar obtaining the sound expected. Additionally, they were able to observe the frequency of the note played, as well as its transformation to a MIDI note in real time.

The mobile application was subjectively evaluated with the feedback obtained from the guitarists, who pointed out the great usefulness of this mobile application because it broadens their possibilities of interpretation with their musical instrument, without requiring more than their mobile device and an audio interface. On the other hand, they mentioned that they noticed a slight latency between the moment they played the guitar and the emulation of the chosen sound, which could be reduced using a low-level programming language. Two examples of the mobile application in operation are shown in Fig. 5 and Fig. 6.

Fig. 5. Example of the Execution of a Musical Note a 440 Hz with the Guitar using the Alto Saxophone Sound through the Mobile Application.

Fig. 6. Example of Playing a Musical Note a 440 Hz with the Guitar using the Acoustic Grand Piano Sound through the Mobile Application.

VI. Discussion

The mobile application was developed with the Java programming language. Due to Java uses a virtual machine (Java Virtual Machine, JVM), which processes the instructions before being executed [19], a considerable latency was obtained in the results. To those who want to return to this article for their research, they are advised to use a lower-level programming language such as C or C++, with which the latency would be reduced considerably and obtain better results, because the instructions are executed directly. Guitar effects pedals commonly use recommended programming languages [20].

VII. Conclusion

The development of mobile applications has acquired great relevance due to the variety of areas where they can be used. As mentioned, there are applications that allow communication between people, planning travel routes, requesting food delivery and entertainment, to name a few examples. Additionally, the applications have allowed people to count with tools that provide the opportunity to explore, grow and develop in other areas such as music, which contributes to their comprehensive training. The mobile application presented in this work benefits this aspect because the user has a synthesizer in the palm of his hand that is capable of emulating monophonic sounds of different musical instruments or other types of sounds with a guitar. This synthesizer, unlike the ones available on the market, is affordable, easily transportable and usable anytime, anywhere. Applications as the one described above allow the practice of music to be accessible to huge number of people.

It has been considered as future work, to make this mobile application a polyphonic synthesizer, where to obtain the polyphony generated by the guitar, the frequency spectrum obtained by the FFT Radix-2 algorithm will be taken up, and search and decision mechanisms will be used. It is also planned to adapt the mobile application to a lower-level programming language like C++ for lower latency.

Acknowledgment

This work was supported by Instituto Politécnico Nacional, COFAA, EDD and SIP. (Project SIP20200832).

References

[1] REAL ACADEMIA ESPAÑOLA, “Diccionario de la lengua española”, 23.ª ed. [Online]. Available: https://dle.rae.es. [Accessed 6 April 2021].

[2] R. Bennett, “Léxico de música”, Madrid: Ediciones Akal S.A., 2003.

[3] Electro-Harmonix, “MEL9”, 2016. [Online]. Available: https://www.ehx.com/products/mel9. [Accessed 25 March 2021].

[4] BOSS, “SY-1000”, 2019. [Online]. Available: https://www.boss.info/mx/products/sy-1000/. [Accessed 25 March 2021].

[5] ROLAND, “GR-55”, 2011. [Online]. Available: https://www.roland.com/mx/products/gr-55/. [Accessed 25 March 2021].

[6] University College Ghent, “TarsosDSP”, 2019. [Online]. Available: https://github.com/JorenSix/TarsosDSP. [Accessed 8 March 2021].

[7] B. Farmer, “Midi Driver”, 2021. [Online]. Available: https://github.com/billthefarmer/mididriver/. [Accessed 8 March 2021].

[8] R. Neuenfeld, M. Fonseca y E. Costa, “Design of optimized radix-2 and radix-4 butterflies from FFT with decimation in time”, 2016 IEEE 7th Latin American Symposium on Circuits & Systems (LASCAS), pp. 171-174, 2016.

[9] J. Strawn, C. Abbott, J. Gordon and P. Greenspun, “The Computer Music Tutorial”, London: The MIT Press, 1996.

[10] R. W. Heath, “Introduction to Wireless Digital Communication”, United States of America: PRENTICE HALL, 2017.

[11] V. Montero, “Software para identificación de música”, Sevilla: Universidad de Sevilla, 2020.

[12] D. Takahashi, “Fast Fourier Transform Algorithms for Parallel Computers”, Japan: Springer, 2019.

[13] J. Jamrich, “New Perspectives on Computer Concepts”, United States of America: Cengage Learning, 2018.

[14] P. Blanchard and D. Volchenkov, “Random Walks and Diffusions on Graphs and Databases An Introduction”, Germany: Springer, 2011.

[15] R. Izhaki, “Mixing audio concepts, practices and tools”, Great Britain: Routledge, 2017.

[16] M. Association, “General MIDI”, [Online]. Available: https://www.midi.org/specifications-old/item/general-midi. [Accessed 6 April 2021].

[17] R. S. Pressman and B. R. Maxim, “Software Engineering: A Practitioner's Approach”, New York: Mc Graw Hill Education, 2019.

[18] R. Mall, “Fundamentals of Software Engineering”, Sonepat: PHI Learning Private Limited, 2018.

[19] Oracle, "Java Virtual Machine Technology", [Online]. Available: https://docs.oracle.com/javase/8/docs/technotes/guides/vm/index.html. [Accessed 6 April 2021].

[20] B. Holmes, “Guitar Effects-Pedal Emulation and Identification”, Belfast: Queen’s University Belfast, 2019.