POLITECNICO DI MILANO - DEPARTMENT OF ELECTRONICS, INFORMATION AND BIOENGINEERING

POLITECNICO
MILANO 1863

Digital Guitar Tuner

[Coding Project]

Student Matheus Fim
ID 876069

Student Lorenzo Binosi
ID 876022

Course Advanced Operating Systems
Academic Year 2016-2017

Advisor Federico Terraneo
Professor William Fornaciari

July 9, 2017

Contents

1 Introduction

The goal of this project was to deliver the code of a guitar tuner precise and fast enough to be satisfactorily used by all kinds of musicians. The main motivation behind the development of the code was enhancing the comprehension of real time operating systems and acquiring programming abilities in this area, such as handling threads, protocols of communication and other algorithms commonly used in operating systems.

1.1 Problem statement

Some minimum requirements for the guitar tuner were accorded by the group in order to guarantee the quality of the project. A satisfactory tuner should deliver consistent measures with a precision of at least 3 hertz. Also, it should be fast enough to give real time feedback of the tuning, more specifically it can not leave the user with the feeling of being slowed down by the hardware.
For the execution, it was used a STM32F407 Discovery board and a single-chip USB-TO-UART Bridge to provide the real time communication. As a starting point it was handled the driver for the microphone embedded in the board. And, from there on, the group had to work towards being able to translate the captured audio into reliable sound frequencies. Those frequencies would then be used to possibilitate the tuning algorithm and a simple user interface.

1.2 Summary of the work

The project had five milestones: Firstly, the driver of the microphone had to be updated to the current version of Miosix; Secondly,the Fast Fourier Transform (FFT) algorithm had to be implemented using Miosix compatible libraries; Thirdly, the initial achieved resolution was of 10.7 hertz, which is far from good for a tuner, so a higer decimation had to be implemented in the microphone; Then, the treatment of wrong measurements, such as harmonics and frequencies out of the curve had to be carried out; and to finalize, a simple user interface to choose the desired tuning and actually execute the work was developed.
As it is expected, the main difficulties the group faced were related to the second and third milestones. In the second, the fft function makes use of specific mathematical functions that needed to be optimized to support the real time requirement of the project while still using the maximum number of samples possible. And in the third, the increase of decimation involved key concepts of both the way the hardware is handled and digital sound.

2 Design and implementation

For the design of the project it was kept in mind the “low coupling and high cohesion“ principle of software design. That being said, it is possible to notice specific files that deals with each part of the tuner. In this way, it is simple to reuse the code in the future and also guarantee an easy maintenance.

The interaction of the user goes from starting the tuner, choosing the desired tuning, and one string at a time, see if he should bring the pitch up or down. When satisfied, the user can press the "Enter" button, on the keyboard, and move forward to the next string. Some cautions are taken, for example, if the frequency received is too far from the one it should be measuring, that probably means the wrong string is being played, and a warning is issued.

Bearing in mind the desired user experience, the flow of the program consists of the microphone‘s driver recording the audio in samplings, and as soon as a buffer is full it is created a new thread that is passed to the FFT function. The function, by its turn, translates the vector of 16-bit PCM audio information into a frequency value. And for the following part, the frequency is compared with the desired one and the feedback is displayed to the user.

In order to discuss what and how was implemented it is of easier comprehension if each functionality is addressed individually. Therefore, starting from the FFT function, it was researched and imported several ARM library functions related to the mathematical operations needed, those functions proved to be optimized, as they were not only orders of magnitude faster than the others available in the literature as they also achieved the goal of execute all the calculations in real time. The FFT function can handle successfully 8192 samples, that is, the 4096 that are provided by the microphone driver plus the imaginary part of each. This initial set up of FFT function and the original microphone driver resulted in a frequency resolution of 10.76 hertz according to the following formula:

$$Resolution = I2SxCLK/[(16 * ((2 * I2SDIV) + ODD) * 8)) * 4096] = 10.76Hz \quad (1)$$

where the sampling frequency is given by:

$$F_S = I2SxCLK/[16 * ((2 * I2SDIV) + ODD) * 8)] = 44.1KHz \quad (2)$$

according to [1].

Moving on to the Microphone driver, it was made the update of the code to make it compatible with latest Miosix kernel. The changes consisted of fixing the callback that due to a difference in the way exceptions were being handle was not working and also renaming functions and variables. Concerning the frequency resolution, it was increased the decimation factor in the PDM filter from 16 to 64 bits, by dropping 3 out of every 4 samples measured. In this way it was achieved a frequency resolution of 2,69 hertz.

To finalize the project, the previously described user interface was implemented concurrently with an algorithm that ignores frequencies that are above a specified delta between the latest measurements, as a form of ignoring background noise and misread harmonics.

3 Experimental evaluation

The experiments were divided into two categories. In the first category, a more technical approach was used and consisted of various frequencies being played by computer speakers and compared to the results given by the tuner, this provided solid information on the reliability and consistency of the measurements. In the second one, aiming real world situations, a classic guitar, an acoustic guitar, an electric guitar and a folk guitar were tuned both with the developed tuner and other tuners sold in the market, providing information on non-functional aspects of the system, as if the response time is adequate.

3.1 Experimental setup

In the first phase, were played all the frequencies that matches the musical notes contained in any of the available tunings, also the frequencies 5 hertz above and below and at least two harmonicas of each note. The frequencies corresponding to each note can be seen in the following tables.

Table 1: E Standard

String	Note	Frequency [Hz]	Harmonics [Hz]
1st	E	329.6	659.2 and 988.8
2nd	B	246.9	493.8 and 740.7
3rd	G	196.0	392.0 and 588.0
4th	D	146.8	293.6 and 440.4
5th	A	110.0	220.0 and 330.0
6th	E	82.4	164.8 and 247.2

Table 2: Eb Standard

String	Note	Frequency [Hz]	Harmonics [Hz]
1st	Eb	311.1	622.2 and 933.3
2nd	Bb	233.1	466.2 and 699.3
3rd	Gb	185.0	370.0 and 555.0
4th	Db	138.6	277.2 and 415.8
5th	Ab	103.8	207.6 and 311.4
6th	Eb	77.8	155.6 and 233.4

Table 3: Drop D

String	Note	Frequency [Hz]	Harmonics [Hz]
1st	E	329.6	659.2 and 988.8
2nd	B	246.9	493.8 and 740.7
3rd	G	196.0	392.0 and 588.0
4th	D	146.8	293.6 and 440.4
5th	A	110.0	220.0 and 330.0
6th	D	73.4	146.8 and 220.2

Table 4: Open E

String	Note	Frequency [Hz]	Harmonics [Hz]
1st	E	329.6	659.2 and 988.8
2nd	B	246.9	493.8 and 740.7
3rd	Ab	207.7	415.4 and 623.1
4th	E	164.8	329.6 and 494.4
5th	B	123.5	247.0 and 370.5
6th	E	82.4	164.8 and 247.2

Table 5: Open G

String	Note	Frequency [Hz]	Harmonics [Hz]
1st	D	293.7	659.2 and 988.8
2nd	B	246.9	493.8 and 740.7
3rd	G	196.0	392.0 and 588.0
4th	D	146.8	293.6 and 440.4
5th	G	98.0	196.0 and 294.0
6th	D	73.4	146.8 and 220.2

Table 6: Open A

String	Note	Frequency [Hz]	Harmonics [Hz]
1st	E	329.6	659.2 and 988.8
2nd	A	220.0	440.0 and 660.0
3rd	E	164.8	329.6 and 494.4
4th	Db	138.6	277.2 and 415.8
5th	A	110.0	220.0 and 330.0
6th	E	82.4	164.8 and 247.2

The experiment was repeated a total of three times, first in a quiet environment, after in a normal one and at last in environment with loud background noise.

For the second phase, the different types of guitar were tuned in all the different tunings in each of the environmental conditions described above.

3.2 Results

For the first phase, the musical notes produced by the computer speakers were correctly recognized by the tuner in the quiet and in the normal environment. It is also capable of recognizing the notes with moderately loud noises in the background, although the correct frequency could not continuously be perceived.

For the second phase, the only guitar that we were not able to tune in any tuning and in any environment was the electric one. This is probably due to the fact that the sound produced by the strings of the electric guitar is really low. Instead, for the other types of guitar, the tuner has been able to recognize all the notes of all the different tunings in the quiet and in the normal environments. Similar results to the first phase has been obtained in the environment with loud noise in the background.

4 Conclusions and Future Works

We succesfully achieved our initial goal of delivering a digital guitar tuner able to help musicians in the tuning of real guitars. Anyway this is not true for electric guitars, but usually, digital guitar tuners, that recognize frequencies via microphone, are not used in the tuning of this type of guitars.

The tunings provided with the tuner are the most used tunings for classic and acoustic guitar. Anyway, musicians could required diffent tunings and a future work could be to implement a function that gives the possibility to the user to set up a custom tuning.

The tuner could be also implemented using a piezoelectric pick up, instead of using the microphone, for a more accurate tuning and also for the tuning of electric guitars. It would required to set up a

pre amplification of the signal, received from the pick up, in order to prepare it for the DAC of the STM32F407 Discovery board.

References

[1] *RM0090 - Reference manual*. STM32F405/415, STM32F407/417, STM32F427/437 and-STM32F429/439 advanced ARM-based 32-bit MCUs.
http://cdn.sparkfun.com/datasheets/Dev/dotNET/DM00031020RM.pdf

[2] *UM1472 - User manual*. Discovery kit with STM32F407VG MCU.
http://www.st.com/content/ccc/resource/technical/document/user_manual/70/fe/4a/3f/e7/e1/4f/7d/DM00039084.pdf/files/DM00039084.pdf/jcr:content/translations/en.DM00039084.pdf