

Treball de Fi de Grau/Màster

Grado en Ingeniería en Tecnologías Industriales

Automatic guitar tuner

Project report

Autor: Antonio Hinojosa Cabrera
Director: Emilio Angulo Navarro
Convocatòria: Enero 2020



Escola Tècnica Superior
d'Enginyeria Industrial de Barcelona



Summary

The present work consists in the design of a guitar tuner system. The proposed system automatically measures the initial musical tone and rotates the tuner system of the instrument to achieve the right tone for the selected string.

As an introduction, the document initially explains the physic connected with the phenomena:

[Wire + Tension] → Induce a Vibration → Generates a Sound → Capture the signal →
Digitalization of the Signal → Compare with a target → Re-Adjust the wire Tension →
Repeat or DONE

- Explain the physical phenomena. How to produce a sound inducing a vibration in a tensioned wire.
- Also explain the mechanical device to tension a wire in a standard guitar.
- Afterwards, we explain the mathematical model used to classify a sound as a musical note.
- It is also important to review all different systems and methods to capture the sound, transform to electrical signal and finally, digitize it. Inside this topic, we also identify PRO's and CON's connected with each of the previous systems.
- The main parameter used to classify the sound is the frequency. After measuring it, we compare with a target and an external stepper motor increment or reduce the wire tension to achieve previous target.
- The existing solutions for tuning musical instruments are mentioned as state of the art.

Subsequently, about the design of the device the project completes these steps:

- System device to capture the sound wave → The selected system was the most suitable one, after evaluating different alternatives. The selection criteria and the rational behind this selection is explained.
- Electrical PCA to capture and amplify the signal before digitizing it. We explain the

two circuit boards designed and prototyped. Each of the EE chips and components used in previous boards, are explained.

- Microcontroller/Arduino. The algorithm to adjust the wire tension depending on the captured sound was programmed in Arduino. We have used Arduino, because it provides high versatility important while prototyping.
- Arduino controller provides two different functionalities:
 1. Determine the frequency for a captured sound signal (mathematic equation).
 2. Control motor movements to re-adjust the wire tension, following a predefined algorithm that minimize the number of iterations.
- Hardware Design. There are three important hardware components:
 1. Stepper Motor. Controlled by the Arduino and to transmit rotation to the guitar tuner.
 2. Guitar Tuner Adaptor. Part to connect and transmit the movement from the motor to the guitar tuner.
 3. Structure Body. To reference all previous devices to the guitar. This part must be adapted to the geometry of the selected guitar.
- Finally, we have estimated the required budget to carry out the project, the environmental impact and the work planning.

Acknowledgement

Foremost, I would like to express my sincere gratitude to my project tutor Emilio Angulo Navarro for the continuous support of my PFG for his patience, motivation, enthusiasm and immense knowledge.

I take this opportunity to record our sincere thanks to Maite Bujaldon for her help and encouragement. I also thanks my parents for there unceasing encouragement and support.

I also place on record, my sense gratitude to one and all who, directly or indirectly, have lent their helping hand in this project.

INDEX

1. INTRODUCTION	9
1.1. Motivation	9
1.2. Objective.....	9
1.3. Scope	10
2. THEORETICAL FRAMEWORK	13
2.1. Sound	13
2.1.1. Pitch.....	13
2.1.2. Loudness	13
2.1.3. Tone.....	14
2.2. Stringed instruments.....	15
2.1.4. Tuner	16
2.3. Frequency range	17
2.4. State of the art.....	19
2.1.1. Tuning fork	20
2.4.1. Conventional tuners	21
2.4.2. Needle tuner	21
2.4.3. Clamp tuner	22
2.4.4. Automatic tuner.....	23
2.5. Basic specifications	25
2.6. Fourier transform	26
2.7. Sampling method	28
2.8. Automatic control	29
2.9. State of technological art.....	29
3. DEVICE DESIGN	33
2.10. Capture interface selection.....	33
2.11. Amplification of the capture signal.....	35
2.12. Add continuous component.....	35
2.13. Amplifier and offset circuit	36
2.14. Digitization interface	37
2.15. Frequency identification.....	39
2.16. Frequency identification assembly	41
2.17. Motor	42
2.17.1. Motor chosen	47
2.17.2. Motor adapter for the tuner.....	48

2.17.3 Motor inclusion in the tuning device	49
4. STRUCTURE DESIGN	53
5.RESULTS	56
a. Assembly	56
b. Motor test	57
c. Guitar tuning	57
6.PLANNING	59
7. ENVIRONMENTAL IMPACT	60
8. BUDGET	61
9. CONCLUSIONS	64
10. BIBLIOGRAPHY	66
Bibliography references	66
Complementary bibliography	66
11. ANNEX	68

1. Introduction

When we use a stringed instrument, it is critical to ensure a right calibration of the wires. When a string vibrates produces a specific musical note, for some reasons (the natural deterioration of the strings, the prolonged use of the instrument or others), the tension of the strings changes, resulting in a different vibration to the desired one, which produces an incorrect musical note. In other words, the string goes out of tune and the user must verify that they are correctly tuned each time the instrument is ready to be used.

1.1. Motivation

There are several solutions to capture the sound provided by a certain string. It is easy to find those devices in music stores. However, these solutions only measure the note that produces the wire of the instrument, afterwards, the musician must adjust manually the tuner to get a new register a converge manually to the target, repeating the process several times.

In this project the electro-mechanic system is able to:

- Capture and digitalize the sound provided by a certain string.
- Compare automatically with a target.
- Automatically adjust, by rotating the tuner the wire tension.
- Repeating the process with an intelligent algorithm that minimizes the number of iterations.

No musician expertise is required in the operator. The only requirement is to correctly assembly the device on the guitar and follow the instructions to generate a wire vibration when required. The adjustment process is automatically executed by the electro-mechanic device.

1.2. Objective

This project includes:

- Design and specifications: mechanical, electrical, firmware with Arduino.
- Prototype Construction: 3D Printed parts, Motors, Sound capturing boards, Arduino...
- Validation and test results with a real case.

Previous prototype solves the problem of tuning a string instrument without musician knowledge, quickly and efficiently.

The device must be able to analyze the sound provided by a wire, compare it with a target, adjust the tension of each string of the instrument and finally deliver it in perfect conditions for using it.

It is proposed as a general objective build a prototype that can be used and works. In other words, that can correctly tune a string instrument.

As secondary objectives are specified in the following points:

- Simplify the mechanical design, without compromising the functionality, this fact will facilitate the prototype manufacturing and construction.
- Test with different solutions to capture the sound in order to decide which one is better for this system.
- Provide a control algorithm that allows tuning the guitar in the shortest possible time.
- The guitar should not lose functionality once the system is installed. In addition, the installation must not require to do modifications in the guitar (machining, drills, etc.).
- Usability while operating the device, which can be used without difficulty with a single hand.
- That works in any guitar, acoustic or electric with the help of a frequency measurer.

1.3 Scope

This work considers the design, construction and tuning of a prototype electromechanical guitar tuner for a string.

The points necessary to achieve this purpose are defined below:

- Introduction to the theory of sound.
- Review the state of the art and study the advantages and disadvantages of each of the most common tuners.
- List of basic system requirements & specifications.

- Study of existing tools to capture and digitize sound.
- Construction of two circuit blocks, an amplifier and a continuous component additive necessary to process the captured signal before digitizing it.
- Programming by Arduino an algorithm that identifies the frequency of the captured signal.
- Control by Arduino a motor that can turn the tuners.
- Use the frequency identification algorithm with the motor programming, in order to tune a string.
- Design a 3D structure as a connection between the guitar and the tuner.
- Integration and assembly the tuner on the guitar.
- Prepare a report about the results obtained.
- Prepare a Gantt chart, showing each phase of the project and its duration.
- Calculation of the cost of the project.
- Study of the environmental impact.

2. Theoretical framework

2.1. Sound

From a physical point of view, the audible sound consists of sound waves that are produced when the oscillations of air pressure are converted into mechanical waves in the human ear and perceived by the brain. The propagation of sound in fluids takes the form of pressure fluctuations.

Sound need a medium of propagation to reach our ears, this is usually the air the propagation speed of the sound in the air is about 343 m / s at 20 C°, and they can reverberate, bounce on different types of surfaces, achieving different effects of echo or distortion, which often magnify their power.

The sound produced by a musical instrument has three characteristics that identify and / or distinguish pitch, loudness and tone.

2.1.1. Pitch

The pitch allows identify a sound as high or low, this characteristic is related on the musical note that perceives the musician. The pitch is related to the amount of times per second that the air is compressed and decompressed, that is, how often the propagation medium vibrates. This is the element that will be studied in the development of this project, because to associate a sound with a specific musical note you must also associate a corresponding frequency. In other words, the process of tuning a musical instrument corresponds to adjusting it to produce a sound with a specific frequency.

2.1.2. Loudness

This characteristic is easily perceived and relate how loud or soft the sound is. Physically, the intensity corresponds to the power emitted or received, or the amount of energy that crosses a surface per unit of time. The most obvious analogy is to associate the loudness of the sound with the volume that the listener is perceiving.

$$I = \frac{A}{N}$$

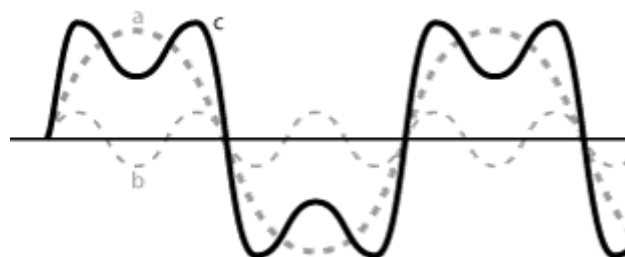
Where “I” is the sound intensity, “A” is the acoustic power and “N” is the area perpendicular to the direction of propagation.

2.1.3. Tone

Feature that distinguishes the source of the sound. Two sounds with the same pitch and loudness, for example, playing the same musical note by a saxophone and a violin at the same loudness, can be discriminated by their tone. Each musical instrument has a characteristic tone. In physical terms, two sounds with the same loudness and fundamental frequency, differ from each other by the harmonics that accompany the fundamental wave.

In other words, it is the combination of harmonic frequencies that defines the tone in a sound.

Therefore, we can classify between simple waves and complex waves, which are composed of two or more simple waves. The following image shows the complex wave indicated with the letter “c”, composed of two simple waves “a” and “b”, the fundamental frequency corresponds to the frequency of the single wave with greater amplitude, in this case the simple wave “a”. The musical note of this complex wave is determined by the fundamental frequency or the frequency of the single wave with more amplitude “a”.



Picture 1. Complex wave

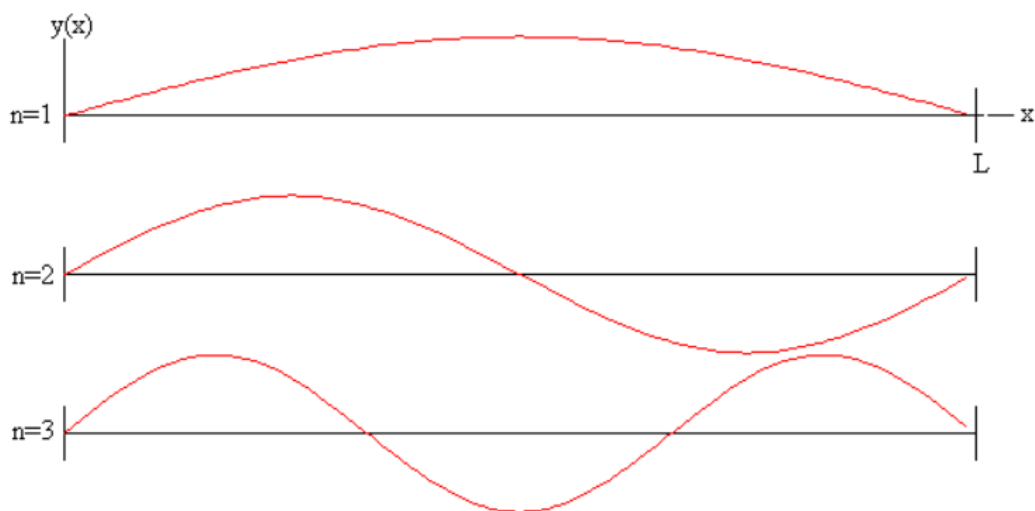
2.2 Stringed instruments

Stringed instruments or chordophones use vibrant strings to produce sound. By vibrating the string, the ends remain fixed and behave like vibration nodes, so it can be studied as a stationary wave. The most important parameter that correlates with the musical note, produces by a vibrating string is the frequency associated to the vibration.

Exists some forms of vibration that generates stationary waves, this are called modes of vibration. The minimum frequency of vibration capable of generating a stationary wave is called the fundamental frequency. When the string vibrates with that frequency it is said that it's the fundamental mode of vibration.

The frequency of the other modes of vibration are multiples of the fundamental frequency and are called harmonics.

The following image shows three vibration modes of a fixed-ends string, the fundamental mode of vibration indicated by $n = 1$ and corresponding harmonics $n = 2$ and $n = 3$.



Picture 2. Vibration modes

In these stationary waves there is a relationship between string length and wavelength $L = \lambda/2$, therefore, considering that the velocity v , is given by the wavelength λ , multiplied by the frequency f .

$$v = \frac{\lambda}{f}$$

We obtain that the fundamental frequency emitted by a string of a vibrating instrument is given by the formula:

$$f = \frac{v}{2L}$$

Where v is the propagation velocity of the wave, corresponding to the frequency of vibration and L is the length of the string. The propagation speed v of a wave in a string is proportional to the square root of the string tension τ and inversely proportional to the square root of the linear mass density δ of the string.

The fundamental frequency of the sound produced by a vibrating string of a musical instrument is given by:

$$f = \frac{1}{2L} \times \sqrt{\frac{\tau}{\delta}}$$

By rotating the tuner of a string instrument, the variables τ and δ are being adjusted. When the tuner tightens the string, less mass is obtained in the same distance, it decreases δ , increases the tension and a higher frequency sound will be obtained. The tuning of this type of instruments consists in applying the correct tension to the string to obtain the desired frequency (musical note).

2.1.4. Tuner

In a musical instrument, the tuner is a piece that is rotated to tighten or tune string. This work is framed in the context of musical instruments whose tuners make use of the worm-crown system. In the case of musical instruments that use this type of gears on their headstock, the crown is attached to the cylinder in which the string is wound. By turning the screw, the movement is transmitted to the crown which rotates the cylinder and cause a change in tension in the string.

The direction of rotation that need to apply to adjust the tension depends on the stringing technique used, but the international convention exposes that by applying a positive torque on the tuner (movement to "screw") the string decrease its tension, and when applying a negative torque (movement to "unscrew") the string increases its tension.



Picture 3. Tuner

2.3 Frequency range

A musical note is a sound with a specific frequency. The human ear is not able to perceive an infinite range of frequencies, only vibrations that produce more than 16 cycles of compression and decompression of air per second 16 *Hz* or those below 20,000 cycles per second 20,000 *Hz* are interpreted as sound for a human being. All the musical notes audible are then, within this frequency range.

In music there are 12 fundamental musical notes that become a cyclical repetition by doubling the frequency of the sound that is emitted, two notes form an octave when the frequency of one of the two notes is twice that of the other note.

For example, the musical note *Do* that is used as a reference *Do* center, also called *Do4* has a frequency of 261.63 Hz the next note, *Re*, sounds at 293.66 Hz, then *Mi* has 329.63 Hz and

if the sound frequency is continued you will find more high musical notes, until the musical note that comes after *Si*, is also a *Do* this time, *Do5* and has a frequency of 523.25 Hz, which is twice the frequency of the central *Do*. It is said then, that this new one belongs to a higher octave. Within the range of human hearing, we can find several octaves of musical notes, each with the same 12 notes.

There is an equation that calculate to each musical note the frequency. Each frequency of the 12 notes belonging to the 10 octaves, responds to the following formula, which "o" corresponds to the octave in which the note is found and "n" is the musical note:

$$f(n, o) = 440 + 220^{(o-14) \times 12 + (n-10)}$$

Therefore, when increases the frequency spectrum, the distance between one note and another is greater. However, the human ear perceives these differences logarithmically. For example, the frequency difference between the *La* and *La #* notes of the fourth octave is 26.2 Hz, while the difference between the *La* and *La #* notes of the fifth octave is 52.3 Hz (double). The difference between *La* and *La #* of the sixth octave will be 104.7 Hz (four times the difference in the fourth octave). However, the human ear perceives the difference between a *La* and a *La #* (and also the difference between any other pair of continuous notes) is the same in any of the octaves.

During the tuning of the guitar, a specific note is assigned to each string, from the first string corresponding to a *Mi* of the 2 octaves to the sixth string corresponding to a *Mi* of the 4 octaves. The frequency range during guitar tuning is 164 Hz to 659 Hz.

The following table shows the notes and their corresponding frequencies of each of the guitar strings.

String	Note	Octave	Frequency
1	Mi	2	82 Hz
2	La	2	110 Hz
3	Re	3	147 Hz
4	Sol	3	196 Hz
5	Si	3	247 Hz
6	Mi	4	330 Hz

Table 1. Frequencies during guitar tuning

2.4 State of the art

A tuner is an assistive device capable of identifying the note produced by the musical instrument. It is commonly used as a reference to know if the musical instrument is out of tune.

The cheapest tuners are also the simplest, they usually use LED lights to indicate if the sound that the instrument is emitting is deeper, higher or if it is at the desired frequency. A good tuner is the one that indicates more precisely the difference between the emitted frequency and the desired frequency.

Most tuners incorporate an electronic circuit that detects the tone that is being emitted and compares it with the desired frequency, some tuners also include a device that allows the user to listen to a sound with the desired frequency and to be able to tune their instrument

"by ear "

the majority of the existing solutions in the market are limited to only detecting the frequency of the sound emitted by the musical instrument and comparing it with the frequency of a desired musical note, that is, it is the user who must finally have to adjust the instrument. In other words, he is who refines the instrument and not the tuning device.

This work aims to make a contribution in the field of assistance solutions for the tuning of instruments and propose a device that, apart from covering the functionality of current tuners, also performs the mechanical work of tuning the instrument.

2.1.1. Tuning fork

The tuning fork was invented in 1711 by British-born musician John Shore. A tuning fork is an acoustic resonator, generally shaped like a fork and constructed of an elastic metallic material, usually steel.

At being beaten with some other object, a tuning fork starts vibrating and resonates at the frequency of a specific tone, usually musical note La-4, that is to say, 440 Hz vibrate. The sound is "pure", that is, it produces a sound with a single fundamental frequency, free of harmonics.

To tune the guitar, we pick the string to the same note, La-4, that the tuning fork provides, when these two waves have the same amplitude, but slightly different frequencies, this gives rise to a vibration whose amplitude varies with time, when treated of sound waves, these amplitude variations will be perceived as variations of loudness, or what is the same, periodic increases or decreases in intensity, which are called shakes or pulsations.

Then, when tuning a guitar, the musician is aware of these pulsations that indicate that the guitar is tuned, since both strings vibrate with the same frequency.

The price of the tuning fork depends on the quality of it, it ranges from 10€, a Wittner tuning fork and a Hava tuning fork until 50€. With that tuner we can achieve the best tuning quality, although the tuning time is much longer.



Picture 4. Tuning fork

2.4.1 Conventional tuners

A tuner uses a microprocessor (alternative to the electrical circuit) to measure the average frequency of an input signal.

When the user makes the instrument play a sound, the tuner detects it by means of a microphone or an input line (for an electric instrument). Then the tuner shows the difference in frequency that exists between the detected sound and the desired musical note, indicating if the sound is deeper, higher or of equal height to the desired note.

This type of tuner is classified into:

- Needle tuner.
- Clamp tuner.

2.4.2 Needle tuner

The needle tuners have two wave pickup systems depending on the type of guitar to be tuned, by a microphone if the guitar is acoustic or by a Jack input line if the guitar is electric.

Afterwards, the microprocessor calculates the frequency and compares it with the desired frequency, indicating a tuned note when the needle is in an upright position, any digression to the right or to the left will indicate that the measured sound is outside the correct tone.

They can replace the needle with LEDs or an LCD screen, they use the same concept.



Picture 5. Needle tuner

There are many companies responsible for the production of this type of tuners, prices range between 7€ and 12€.

The quality of this tuning system depends on the method of picking up the wave used, a microphone is used in acoustic guitars, therefore, the ambient noise that negatively affects the tuning quality is also captured, while in an electric guitar, Jack input line is used, ambient noise is ignored and excellent tuning quality is achieved.

2.4.3 Clamp tuner

The clamp is a system developed in 1995 by Mark Wilson of the OnBoard Research corporation. This is the interface that the tuner uses to capture the audio signal.

This type of tuners does not use a microphone or a jack input line, but they adjust by pressure to the wood of the instrument, so that the whole device vibrates along with the rest of the instrument when the string of the instrument is pressed. That is to say, the mechanical wave travels from the string, transmitted to the instrument wood and subsequently transmitted to the device itself.

As they do not use microphones, these tuners ignore all the ambient noise that may be in the room in which the instrument is located, allowing them to remain effective in open spaces or in rooms with high noise levels.

The prices of this type of tuner range between 15€ and 30€. The tuning quality is excellent for any type of guitar.



Picture 6. Clamp tuner

2.4.4 Automatic tuner

The automatic tuner is a compact gadget, a robotic battery tuner that refines the instrument in a few seconds, in the tuners seen so far, is the user who adjusts the instrument, instead, this device in addition to tuning adjusts the instrument, therefore, it would be a fully automatic tuning.

There are two variants depending on the position of the tuner on the guitar.

In the first variant the tuner is fixed to the pegbox of the guitar, combines a clamp tuner together with robotic tuners that allow adjusting the tension of the strings, the user must clamp the strings separately and automatically the robotic pegs will adjust the string to get the desired note.

In 2005 the Tronical company installed an automatic tuner called 'Power Tune' on some Gibson brand guitars, the price of the guitar with the automatic tuner is about 2,500€.



Picture 7. Power tune in a Gibson guitar

The automatic tuner 'Power Tune' by separate, has a cost of 100€.



Picture 8. Power tune

Power tune tuner features:

- Device fixed to the guitar head; it is necessary to replace the previous tuners to place those of the device.
- In the installation, the guitar is not modified.
- Valid for almost any guitar model.
- Fast and precise tuning.

The second variant, is a separate device from the guitar, uses a microphone to capture the wave of the guitar and incorporates a crank. To tune the guitar, we must place the device on each of the tuners and pinching the corresponding string at the same time the crank will rotate the tuner to get the desired note.

The Jowoom company sells this tuner for 40 €, the tuning quality is bad due to the

problem of capturing the ambient noise by the microphone, but the tuning time is very fast.



Picture 9. Automatic crank tuner

In summary:

System	Advantages	Disadvantages
Tuning fork	Great tuning quality	High tuning time
Clamp tuner	Low cost	Great tuning quality
Needle tuner	Low cost	For acoustic guitar low tuning quality
Power tune	Great quality and fast tuning	High cost
Automatic crank tuner	High tuning time	Low quality

2.5 Basic specifications

- This section lists the objectives required in the tuner design:
- The automatic tuner installation doesn't require any modification in the guitar.

- The clam system is recommended to get a better more reliable frequency of the vibration and in consequence a perfect tuning.
- Total hardware cost must be lower than 100 €.
- The prototype to be customized for using in a Epiphone guitar with groover tuners.
- The prototype to be customized for using in any of the six tuners in the previous guitar.
- Easy usability:
 1. Hardware device assembly on the guitar could be done by a single user
 2. Automatic fine-tuning process could be executed with one single hand
- Total required time:
 1. Hardware assembly on the guitar must be completed in less than 120 sec.
 2. Automatic fine-tuning process must be executed in less than 60 sec
- The tuning process is simple and intuitive. No musician expertise is required in the operator.
- The design will be as ergonomic as possible, not exceeding twice the dimensions of the headstock.
- Precision of the final tuning: Difference between target and actual frequency less than 5 Hz.

2.6 Fourier transform

To detect frequencies of a signal that has already been captured, there are techniques that perform calculations on the signal values in the time domain. However, in general, it is much more efficient, less computational cost, accurate and simple, use the frequency domain. A captured signal has a certain number of samples that are the values of the signal at different times. To study the signal in

the frequency domain is used the Discrete Fourier Transform (DFT). Fast Fourier Transform (FFT) corresponds to the computational algorithm that allows DFT to be calculated from a discrete time signal.

From a discrete time, signal, the following points are defined:

- f_s , the sampling frequency, the number of samples per time that was used to build the discrete signal.
- T , the sampling period, the duration of the temporary signal.
- N , the total number of samples contained in the discrete signal, that is $N = T \times f_s$
-

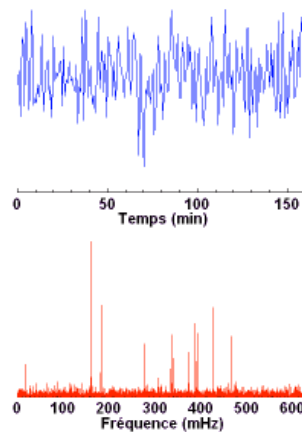
The DFT is calculated from:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn}$$

Which X_n is the n-element of the discrete signal and X_k is the k-element of the DFT of the signal.

To identify the fundamental frequency of the guitar and be able to compare it with the frequency of the desired note, we should observe the largest peak in the frequency spectrum. Other smaller peaks in harmonic frequencies can be observed, these are the other frequencies that accompany the fundamental frequency and give the characteristic timbre to the sound.

The following image shows an example of the frequency spectrum in red, and an input signal in blue. The highest peak in the spectrum of frequencies is at 160 *mHz*, this is the fundamental frequency of the signal from the input signal. The input signal is divided into simple waves, of which the greater amplitude has a frequency of 160 *mHz*.



Picture 10. Frequency spectrum

The Fourier Transform takes a series of framed data over a given period of time and returns the frequency spectrum of the data. The FFT is a computational implementation of the Fourier Transform and therefore, due to the discrete nature of a series of computationally stored data, does not return a continuous spectrum. Then, the transformed signal is also discrete and contains the frequency spectrum information in a finite number of intervals (also called bins) equispaced in the frequency domain.

2.7 Sampling method

In the information theory, there is a fundamental theorem that establishes a condition for an analog signal to be reproduced correctly. It is the Nyquist-Shannon sampling theorem and states that the exact reconstruction of a continuous periodic signal from samples in moments of time is mathematically possible if the signal is limited in frequency band and the sampling rate is higher than double its bandwidth.

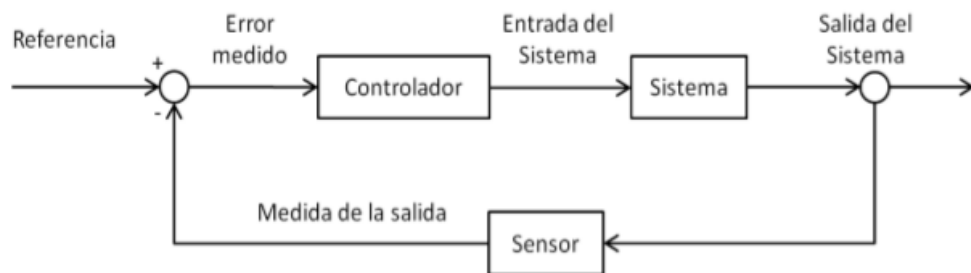
Taken into the context of this work, this theorem states that for rebuilt an analog electrical signal whose fundamental frequency is F , then the sampling rate in the process of digitizing the signal f must be twice the maximum frequency that is being sampling or wanting to recover:

$$f \geq 2F$$

2.8 Automatic control

Any physical system, whose state changes over time, is called a dynamic system. A musical instrument is a dynamic system, the system variable that is of interest, in this case, would be the frequency which a particular string of the instrument vibrates.

There is a mathematics and engineering discipline called automatic control, which studies the behavior of dynamic systems, and states that when it is desired that a certain variable in a system follow a certain reference value, it is possible to build a controller, that monitors some element of the system to achieve that the studied variable approaches the desired value. In this work, the controlled system is the musical instrument, the element that is controlled is the tension of the string, and the variable studied is the frequency of vibration of the string.



Picture 11. Loop control

Image 11 shows a scheme called a loop control. There is a reference, which generally corresponds to a desired value for the study variable. When the output variable of the controlled system moves away from the reference in time, a

controller manipulates the system input to bring the system output back to the reference (feedback). The feedback can be negative (self-compensatory regulation) or positive ("snowball" or "vicious circle" effect).

2.9 State of technological art.

In this section, are studied the possible alternatives on the components or technologies for

the design of the prototype.

The microcontrollers present in the market will be studied, indicating the advantages and disadvantages of using each of them in the project.

Many of the engineering projects are related to research, prototypes and tests, generally the most practical solution involves using a microcontroller, this device used not only in the control of electronic devices but also in the monitoring of processes.

One of the requirements of these devices is that they cover many technological aspects, have a low cost, that they have multiple inputs and outputs and their tools for implementation are also simple.

This project requires a microcontroller that has:

- more than six input and output pins.
- The minimum sampling frequency is 990 *Hz*, in order to identify musical notes, in the case of the guitar, according to the Nyquist theorem seen in point 2.7 is twice the frequency of the string, in a worse case the sixth string vibrates tuned to 330 *Hz*, therefore the sampling frequency must be greater than 990 *Hz*
- Simple programming.
- Low cost.

Based on these four points, the Arduino, Raspberry pi and Beagle bone systems have been studied. The programming of frequency identification and motor control has been studied in the programming module.

System	Pins	Sample frequency	Programation	Cost
Arduino Uno	16	8.928 kHz	Impossibility of performing the Fourier transform to identify the frequency. Simple motor control.	8 €
Raspberry pi	21	18 kHz	Possibility of performing the fast Fourier transform to identify the frequency. Simple motor control	46 €
Beagle bone	65	23 kHz	Possibility of performing the fast Fourier transform to identify the frequency. Simple motor control	48 €

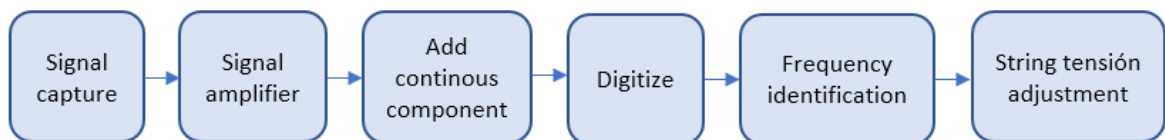
Table 2. Microcontroller comparison

Arduino Uno is the most used microcontroller, this entails a greater number of documents related to Arduino programming that facilitate the learning of the program, as well, as a greater possibility of reusing algorithms and libraries. The frequency identification will be done by a different method than the Fourier transform.

Arduino is selected as the project's microcontroller because of the ease of programming and its low cost.

3. Device design

The operation of the device that proposes this work goes hand in hand with its design and both can be expressed in blocks or modules with different functions:



Picture 12. Block diagram of device operations

Image 12 shows a diagram block representing each of the modules that are part of the electromechanical tuner. Each of these modules will be designed independently according to the task they fulfill in the complete tuning process.

2.10 Capture interface selection

The first point indicates that the first stage in the tuner operation process is to acquire the acoustic signal emitted by the instrument to be tuned. Within the design of the device, the first stage will also be to choose an appropriate interface to convert the acoustic signal into an electrical response that can be studied, treated and from which the device can make decisions. Three important criteria are then established to choose a good capture interface:

- Low noise level: it is necessary to have a signal that does not bring unwanted information, that is to say, the interface must be able to transform only the sound emitted by the instrument into an electrical response and not that of other sources, which, for effects of the tuner operation, are considered noise. In order to capture the sound as clearly as possible and achieve precise tuning of the instrument, the input signal must be around 10 dB higher than the noise.
- Low cost: there are many different types of interfaces. Those that perform a more faithful replica (higher quality) of the transformed signal are considerably more expensive. For the purposes of tuning an instrument, it is only necessary to be

able to recover correctly the frequency of the captured signal and not have a professional studio quality capture.

- Reduced size: this criterion is valid for all elements of the tuner and particularly for the interface; it is necessary that it is not very invasive and does not exceed the natural dimensions of the parts of the instrument to be tuned.
- According to these required characteristics we opt for a contact microphone, this type of microphone is ideal for the application that is being proposed, because although it does not have a high quality of audio in the signal obtained, the important information (frequency) is recovered with no error. The mechanical wave that produces the string is propagated by the wood of the musical instrument towards the piezoelectric material of the contact microphone. In this way, any other acoustic sound or noise present in the room in which the instrument to be tuned is being used, is simply not detected (due to the absence of a diaphragm) by the contact microphone.



Picture 13. Korg cm200

This microphone meets all the exposed criteria:

- It is not sensitive to noise. Because it has no diaphragm, it does not produce an electrical response to sounds that spread through the air.

- It is economical and easy to obtain.
- Its size is limited to the clamp that affirms the device to the instrument and its most important dimension does not exceed 30 mm.

2.11 Amplification of the capture signal

The first problem that arises when using a passive microphone directly, is that the electrical signal that is delivered is too weak or with too low amplitude. In order to be treated, the analog signal produced by the microphone will be digitized.

Microcontrollers and digital data processing systems generally have a dynamic range from 0 V to 5 V. This means that an integer between 0 and 5 is assigned an integer between 0 and 255 (if it has a resolution of 8 bits), distributed linearly. Thus, for example, if a microcontroller detects a voltage of 0 V, the number 0 will be assigned. If a voltage of 5 V is detected, the number 255 will be assigned.

If a voltage of 2.5 V is detected, then the number 127 will be assigned. Any value that is below 0 V is assigned the number 0 and any value

that is above 5 V is assigned the number 255.

The problem of having a very weak or low amplitude analog signal is that resolution is lost in the information obtained when sampling is performed.

In order to use all the integers available by the controller, it is necessary to amplify the signal to a maximum of 5 V. Then, before starting the process of digitizing the signal, the tuning device must have an amplifier to leave the analog response of the microphone in the correct range that the microcontroller will use.

2.12 Add continuous component

It was already said that the use of an amplifier corrects the problem of wasting the processing capacity of a microcontroller, but there is still the problem that the signal is being represented "in the middle". The analog signal oscillates around the 0 V value even after being amplified and all negative voltage values that are detected in the scanning process are

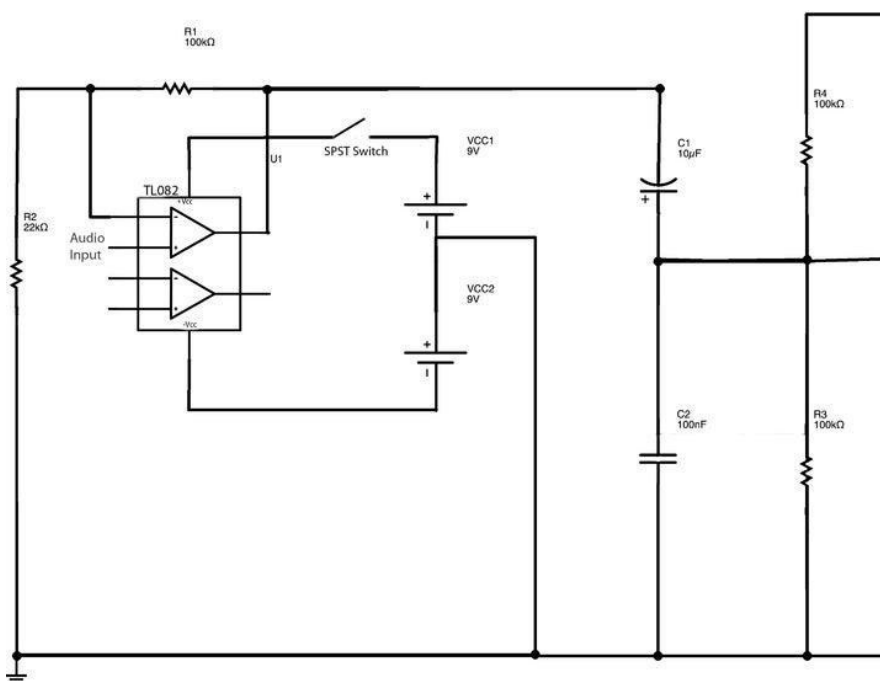
ignored. To solve this, it is necessary to add a continuous 2.5 V component to the signal that has already been amplified, so that the central value is not 0 V but 2.5V.

If the amplified signal is shifted “up” in 2.5 V then all negative values that were previously ignored in the scanning process, this time will be detected, since they will be values between 0 V and 5 V.

An offset circuit is then proposed, using two resistors, two condensers and a 9V voltage source.

2.13 Amplifier and offset circuit

The following image shows the scheme of the amplifier and offset circuit. This circuit has been designed by Amanda Ghassaei and presented in a tutorial about the audio input in Arduino. The scheme created by Amanda Ghassaei comes from the article on Arduino frequency detection from the following link: <https://www.instructables.com/id/Arduino-Frequency-Detection/>.



Picture 14. Amplifier and offset scheme

This circuit integrates:

- A non-inverting amplifier, using two resistors, an OPAMP and a 9V battery. This block amplifies the signal to 5 V.
- Offset, using two resistors and two capacitors, the effect of this circuit block on the analog signal is to add a continuous 2.5 V component, in other words, the analog signal is moved 2.5 V “up”. At the input of this block the amplified signal is sent and at the output a signal, with the same information, is recovered, but, instead of oscillating between -2.5 and 2.5 V, the range of voltages reached varies between 0 and 5 V.

2.14 Digitization interface

The microcontroller used in the design of the tuning device is Arduino Uno, the choice of this driver is due to the following points:

- It has an infinity of algorithms that we can reuse, thanks to being one of the most popular protocols.
- Programming is simple and intuitive.
- Low cost.
- It has a sampling frequency greater than 990 Hz.

Arduino is a free hardware platform, based on a board with a microcontroller and a development environment, designed to facilitate the use of electronics in multidisciplinary projects. The Arduino board is an interface that leaves the digital inputs and outputs of the microcontroller available and allows the microcontroller to be programmed through the Arduino development environment, based on the C programming language.




Picture 15. Arduino Uno

Arduino controller programming is based on two main and necessary functions, the void setup () and the void loop ().

- The setup function is the first function to be executed. Some criteria are established that require a single execution, such as the declaration of variables, to configure the Arduino pins to output or input, and initialize the serial communication. In short, it constitutes the preparation of the program.
- The loop function in Arduino is the one that runs an infinite number of times. When the Arduino is turned on, the setup code is executed and then the loop is entered, which is repeated indefinitely until the microcontroller is turned off or restarted. It contains the code that will be executed continuously (reading inputs, activating outputs, etc.)

With the void loop () and void setup () functions, we give the instructions to the Arduino microcontroller. Everything inside the setup block will be executed once. The contents of the loop block will be executed in the loop while the Arduino controller remains on.

A screenshot of an IDE window showing a code editor. The window title is 'ANTONIO_EXAMPLE_jan'. There are two tabs: 'ANTONIO_EXAMPLE_jan' and 'ReadMe.adoc'. The code in the editor is as follows:

```
1  /*
2
3  */
4
5  void setup() {
6      ...
7  }
8
9  void loop() {
10     ...
11 }
12
```

Picture 16. Arduino primary functions

2.15 Frequency identification

So far, the problems of capturing the sound of the musical instrument have been dressed; treating the captured signal to prepare it for the scanning process and adjusting the microcontroller with an adequate sampling frequency that allows identifying musical notes according to their frequency spectrum (digitization). In other words, the design is covered up to the first 4 blocks. The next step is the identification of the fundamental frequency of the digitized signal.

The tuning device proposed in this title work must adjust the string tension of the musical instrument until a specific frequency sound is produced. Before making the decision to move the tuner and determine in what sense or how much to move it is necessary to have a reading of what is the frequency that is occurring before tuning. In this section we will attack the concept that involves the fifth block.

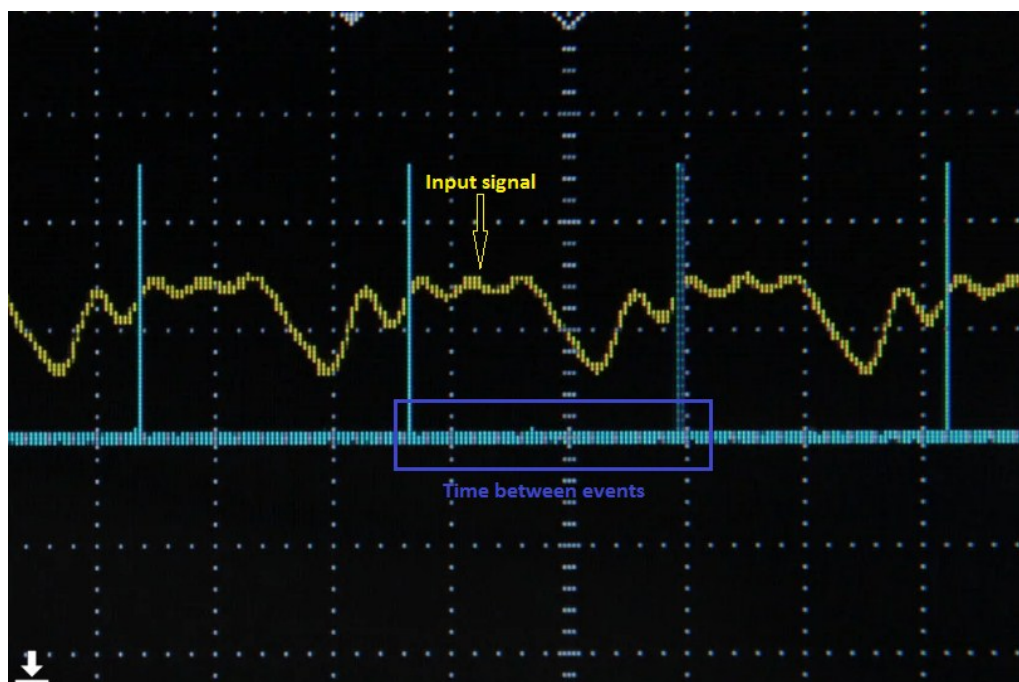
Identifying the frequency of the string would be simple by applying the Fourier transform, unfortunately the controller device used, Arduino Uno, lacks of memory and power to perform this operation. Therefore, an alternative has been searched to identify the frequency.

The algorithm created by Amanda Ghassaei is used in the article on Arduino frequency stop of the following link: <https://www.instructables.com/id/Arduino-Frequency-Detection/>. This code together with the amplifier and offset circuit of section 3.5 allows to calculate the frequency coming from the guitar in Arduino.

As explained before, the acoustic wave coming from the guitar is modified until we obtain an electric wave centered at 2.5 V and between 0V to 5V.

The algorithm for calculating the Amanda frequency is based on identifying when the wave crosses 2.5V with a positive slope (an event is generated), keeping its slope and operating a stopwatch, this algorithm is restarted every time it finds a higher slope, and returning the time passed when it finds a coincidence between slopes.

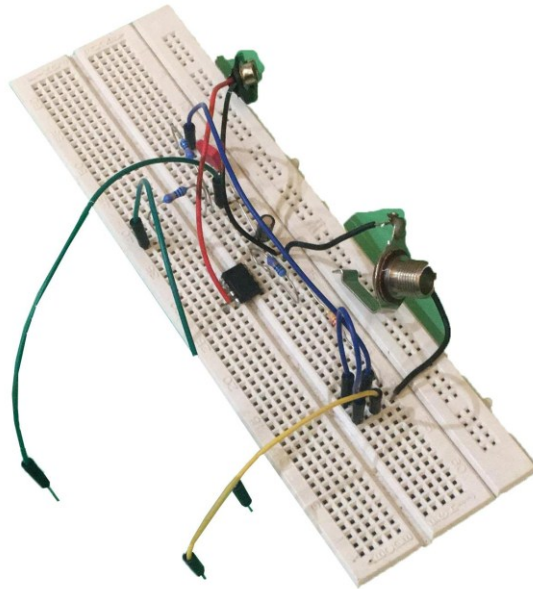
The following image shows the operation of the algorithm on an input signal, the time between events indicates the time passed by the wave to reach the same point with the same slope, that is, the time it takes to repeat the wave. The input signal is a complex wave that can be separated into simple sinusoidal waves, the larger wave governs the frequency of the complex wave, therefore the frequency between events coincides with the fundamental frequency of the input signal.



Picture 16. Frequency identification

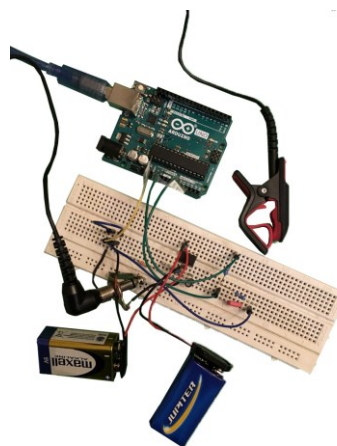
2.16 Frequency identification assembly

The circuit assembly seen in section 3.4 is as follows:



Picture 17. Amplifier and offset assembly

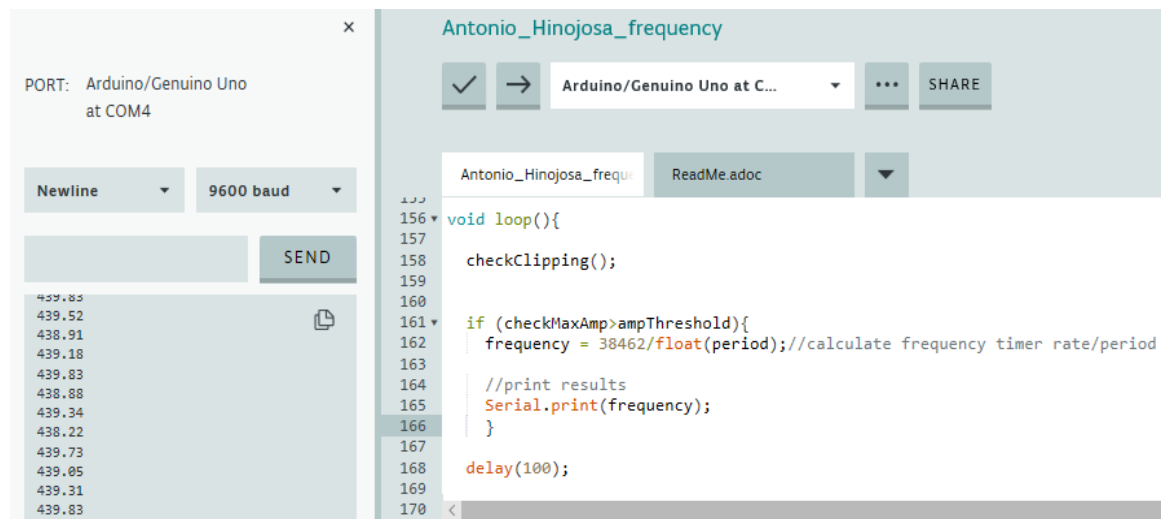
The complete assembly, (image 18) to identify the frequency is done by connecting the output of the circuit to one of the inputs of the Arduino microcontroller and loading the algorithm seen in section 3.6 in the Arduino.



Picture 18. Amplifier, offset, microphone and Arduino assembly

The function of Arduino Serial print and println, allows to write characters in the Serial port at the left column of the browser, now we can visualize the frequency measurements and check the correct operation by identifying the frequency.

In this test the La note has been tried in the fourth octave with an approximate frequency of 440 Hz.



The screenshot shows the Arduino IDE interface. On the left, the serial monitor displays a list of frequency values: 439.83, 439.52, 438.91, 439.18, 439.83, 438.88, 439.34, 438.22, 439.73, 439.05, 439.31, and 439.83. The serial port is set to 'Arduino/Genuino Uno at COM4' with a baud rate of '9600 baud'. On the right, the code editor shows the following code:

```

Antonio_Hinojosa_frequency
✓ → Arduino/Genuino Uno at C... SHARE
Antonio_Hinojosa_frequ ReadMe.adoc
156 void loop(){
157
158   checkClipping();
159
160
161   if (checkMaxAmp>ampThreshold){
162     frequency = 38462/float(period);//calculate frequency timer rate/period
163
164     //print results
165     Serial.print(frequency);
166   }
167
168   delay(100);
169
170

```

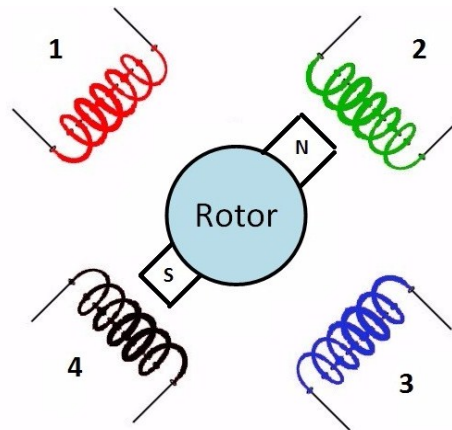
Picture 19. Frequency identification code for Arduino

2.17 Motor

The motor chosen is a unipolar stepper motor because:

- Greater precision in the position and repetition of the movements, we do not need an external position sensor (encoder).
- In continuous drives: this is the case of applications that use long periods of work, where there are many stops and arrangements. Stepper motors offer greater accuracy in these work environments.

A unipolar motor, has 4 coils, can be operated in three different ways:



Picture 20. Stepper schematic

- Normal movement: It consists of exciting, administering current, to two coils at the same time in each step. The maximum torque is achieved but it is also the maximum consumption.

STEP	COIL 1	COIL 2	COIL 3	COIL 4
1	HIGH	HIGH	LOW	LOW
2	LOW	HIGH	HIGH	LOW
3	LOW	LOW	HIGH	HIGH
4	HIGH	LOW	LOW	HIGH

Table 3. Normal movement sequence in stepper

- Motion by wave or full step: It consists of exciting one coil at a time. The consumption is reduced, but also the torque, therefore, is a moderate consumption and torque.

STEP	COIL 1	COIL 2	COIL 3	COIL 4
1	HIGH	LOW	LOW	LOW
2	LOW	HIGH	LOW	LOW
3	LOW	LOW	HIGH	LOW
4	LOW	LOW	LOW	HIGH

Table 4. Motion sequence by wave in stepper

- Half-step movement: A slow and smooth movement is achieved with a torque and consumption average of the other two movements.

STEP	COIL 1	COIL 2	COIL 3	COIL 4
1	HIGH	LOW	LOW	LOW
2	HIGH	HIGH	LOW	LOW
3	LOW	HIGH	LOW	LOW
4	LOW	HIGH	HIGH	LOW
5	LOW	LOW	HIGH	LOW
6	LOW	LOW	HIGH	HIGH
7	LOW	LOW	LOW	HIGH
8	HIGH	LOW	LOW	HIGH

Table 5. Stepper half step motion sequence

The first way to set our motor is manually, in the Arduino environment the coils are assigned into the digital outputs to control the actuation of each one, the algorithm is coded according to the order of movement that want to use. The following image shows the code to get a step



with normal movement.

```
HORARIO_MOTOR.ino  ReadMe.doc  ▼
1 #include <Stepper.h>
2
3 int motorPin1 = 8;
4 int motorPin2 = 9;
5 int motorPin3 = 10;
6 int motorPin4 = 11;
7
8
9
10 void setup()
11 {
12   Serial.begin(9600);
13   pinMode(motorPin1, OUTPUT);
14   pinMode(motorPin2, OUTPUT);
15   pinMode(motorPin3, OUTPUT);
16   pinMode(motorPin4, OUTPUT);
17
18   digitalWrite(motorPin1, LOW);
19   digitalWrite(motorPin2, LOW);
20   digitalWrite(motorPin3, LOW);
21   digitalWrite(motorPin4, LOW);
22 }
23
24 void loop()
25 {
26
27   digitalWrite(motorPin1, LOW);
28   digitalWrite(motorPin2, LOW);
29   digitalWrite(motorPin3, HIGH);
30   digitalWrite(motorPin4, HIGH);
31   delay(10);
32
33   digitalWrite(motorPin1, HIGH);
34   digitalWrite(motorPin2, LOW);
35   digitalWrite(motorPin3, LOW);
36   digitalWrite(motorPin4, HIGH);
37   delay(10);
38
39   digitalWrite(motorPin1, HIGH);
40   digitalWrite(motorPin2, HIGH);
41   digitalWrite(motorPin3, LOW);
42   digitalWrite(motorPin4, LOW);
43   delay(10);
44 |
45   digitalWrite(motorPin1, LOW);
46   digitalWrite(motorPin2, HIGH);
47   digitalWrite(motorPin3, HIGH);
48   digitalWrite(motorPin4, LOW);
49   delay(10);
50
51
```

Picture 21. Coding a step for a normal movement

Using the for () function, we would be able to repeat the previous code n steps.

The other way to program a stepper motor is to use the stepper library that is included with the official Arduino development environment, which uses a wave motion. This library facilitates the use of this type of motors.

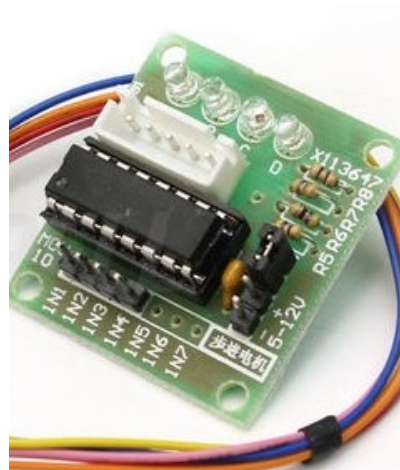
First import the stepper library, add the steps per revolution in the STEPS variable, the number of the pins connected to the four motor coils, and the desired motor speed, finally move the motor a certain number of steps through the numsteps variable, direction schedule with a positive and counterclockwise value with a negative one.

```
1
2 #include <Stepper.h>
3
4
5
6 Stepper stepper(steps, 8, 9, 10, 11);
7
8 void setup() {
9
10     stepper.setSpeed(5);
11 }
12
13 void loop() {
14
15     stepper.step(numsteps);
16     delay(1000);
17 }
```

Picture 22. Stepper library code

To control the stepper motor by Arduino it is necessary to use a driver motor, an Arduino pin can only values of 0 and 5 volts, and give up to 40 mA of current. This is insufficient to move almost any motor.

A motor driver is a current amplifier, whose function is to take a small low current control signal and convert it into a high current signal that can power the motor. There are many types of motor drivers depending on the motor to be driven, maximum voltage, maximum output current, etc



Picture 23 ULN2003

In this project the normal movement is the most suitable to drive the stepper motor, by prioritizing the use of a small motor, these lack a high torque and hinders the objective of

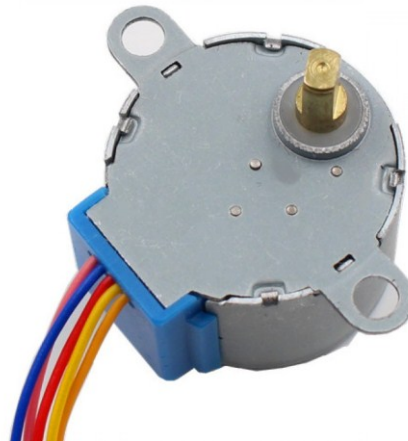
turning the tuner, the motor is programmed manually, since the stepper library uses a wave movement.

2.17.1 Motor chosen

The choice of the stepper motor will depend on the maximum torque generated, the function of the motor is to turn the tuner to tighten or loosen the string and thus tune the guitar, therefore, the motor torque must be greater than the torque needed to rotate the tuner.

A torque measurement test of the tuner was made to select the motor. To rotate the tuner a mass of 700gr is needed at the end of the tuner, it would be equivalent to a force of 6.8 Newton, the distance between the point of application of the force and the axis of the tuner is 1 centimeter, therefore, the moment necessary to turn the pin is 0.068 N*m.

According to the exposed requirements of maximum size and torque, the motor chosen is the 28BYJ-48 stepper motor.



Picture 24. Stepper 28BYJ.48

The stepper motor incorporates a motor driver with the following characteristics:

- Four LEDs that indicate when a coil is excited.
- ULN2003 chip containing three Darlington transistors, which performs the function of current amplifier.
- Four inputs for the controller, referring to the four motor coils.

- Jumpers to select the operating voltage, 5V or 12V.



Picture 25. Stepper 28BYJ-48 and motor driver.

The number of steps required to rotate the motor shaft 360° depends on the motor and the gearbox, in this case the motor has 64 steps and the gearbox is 1:64, therefore 4,096 steps are needed to rotate the shaft 360°.

2.17.2 Motor adapter for the tuner

Finally, this section cover the stage indicated in the sixth point, after being able to control the motor, it is necessary to design the mechanical interface that will link the movement of the motor's rotation axis, with the movement of the tuner of the musical instrument to be tuned, in this case the guitar.

To allow the motor to transmit its torque and movement on the tuner of the musical instrument, a negative piece of the tuner was designed, this piece was fixed to the axis of the motor. In this way, the design of the mechanical interface that can interact with the instrument to be tuned is covered.



Picture 26. Motor adapter for guitar tuner

The adapter design was done by the Creo parametric CAD program and printed in 3D.

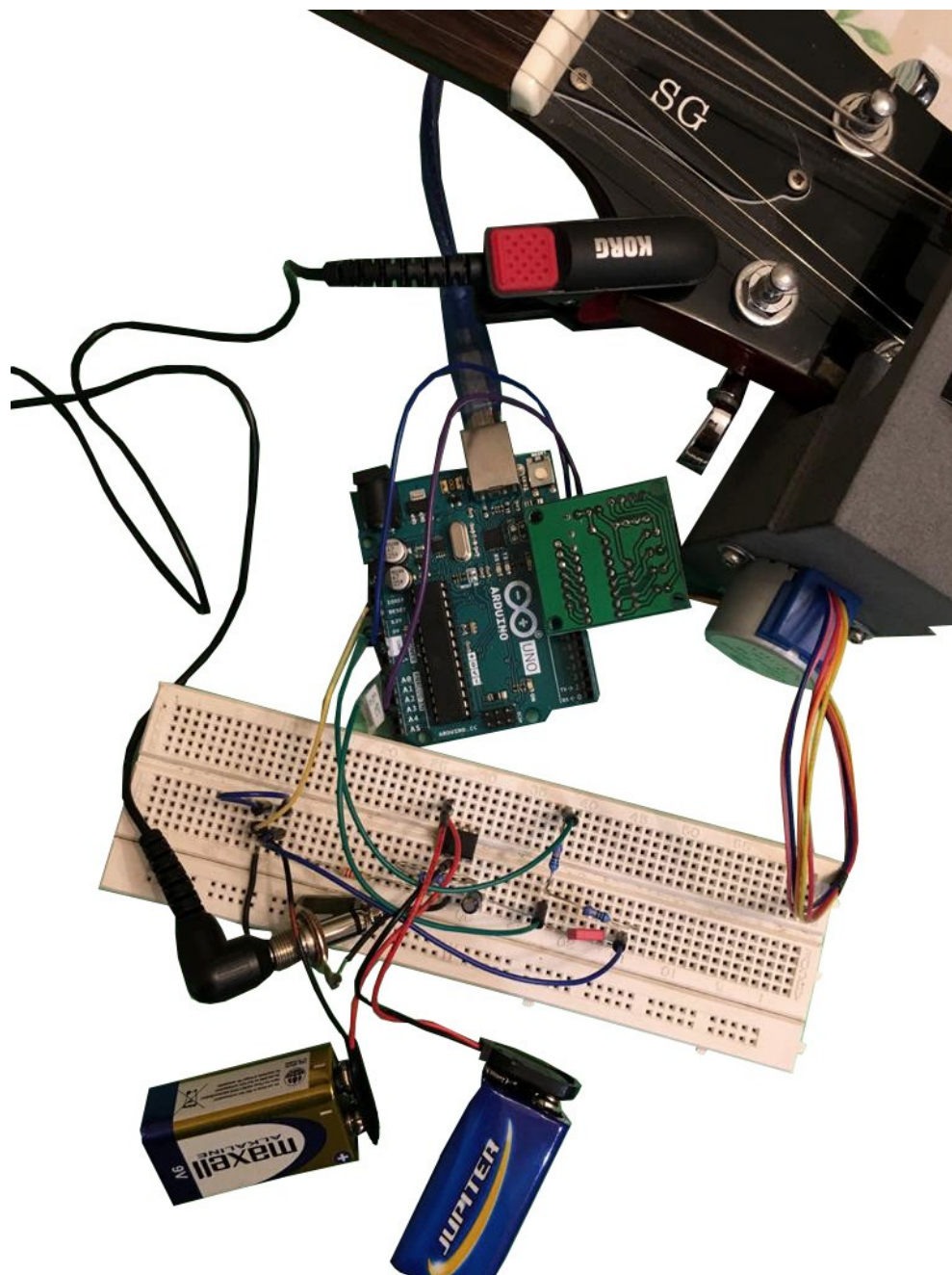
2.17.3 Motor inclusion in the tuning device

It is now necessary to control the motor based on the measured frequency. The speed and direction of rotation of the motor must be controlled by the difference between the frequency of the musical note, corresponding to the string being tuned, and the frequency measured. To achieve this, a simple mathematical function is implemented in the algorithm code, this function is called `freq ()` and receives as parameters two integers corresponding to the note and the octave in which want to calculate the frequency.

```
34  
35 float freq (int n, int o)  
36 return 440.0*pow(2.0,(1.0*(o-4)*12+(n-10))/12.0);  
37  
38
```

Picture 27. Frequency equation code

The next step is to add the motor to the complete assembly, including the circuit proposed in section 3.4. The assembly is as follows:



Picture 28. Complete circuit assembly

The code in section 3.7 define the calculation of the frequency on the captured signal. Now, from that value, it is necessary to make use of the motor code, seen in section 3.9, to control the motor rotation. The code shown in the annex code identify the frequency of the string and rotate the motor axis a determined angle, according to the difference between the desired and the measurement frequency on the guitar.

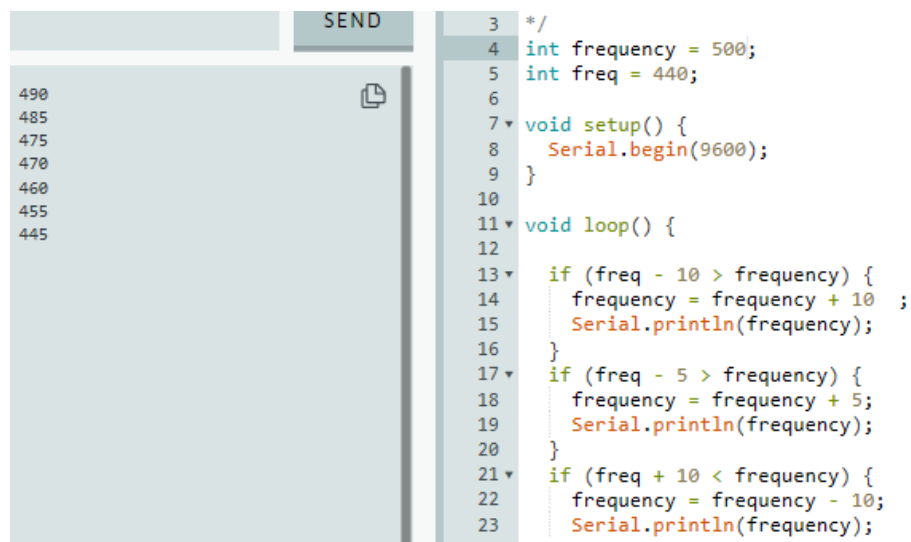
In addition, two intervals are defined, when the measured frequency differs by 10 Hz at the desired frequency and when the measured frequency differs by 5 Hz at the desired frequency.

These intervals define a range of rotation of the motor, the relationship between the rotation of the tuners and variation of the tone is studied. Rotating the tuner more than a quarter turn means a loss of control in the tuning and rotating the tuner an eighth turn is the minimum turn that varies the frequency of the string.

The following relationship is defined: when the frequency differs by 10 Hz, the motor rotates a quarter turn and if the difference is 5 Hz, the motor rotates an eighth turn, in that way we reduce the tuning time.

The following image shows an approximate behavior of the tuner, the table on the left shows the measurements of the frequencies of the string each time we operate the motor.

Initially the value of the frequency is 500 Hz, the desired frequency is 440 Hz, we assume that rotating the tuner 360 degrees means a 40 Hz change in the frequency. We observe that due to the variable angle of rotation as a function of the frequency we approach faster to the correct frequency, in addition the algorithm ends when the identified frequency approaches 5 Hz or less of the desired frequency.



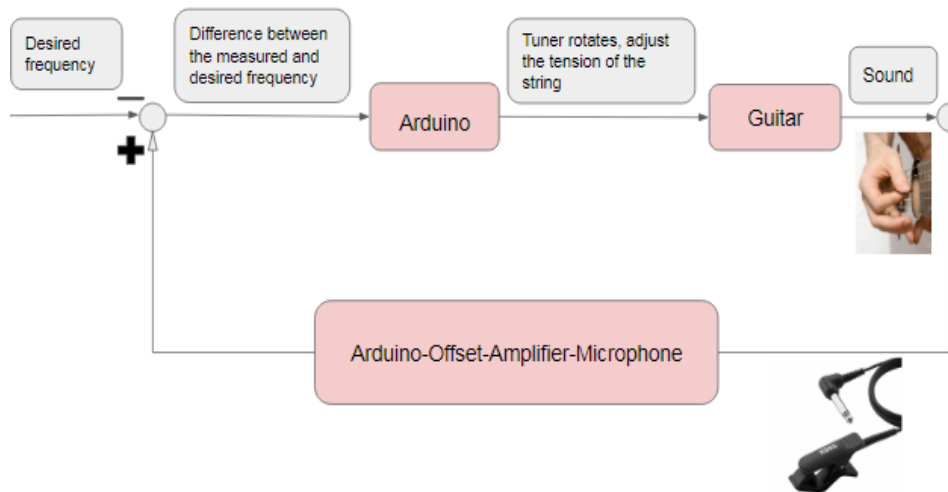
```
SEND
490
485
475
470
460
455
445

3 */
4 int frequency = 500;
5 int freq = 440;
6
7 void setup() {
8   Serial.begin(9600);
9 }
10
11 void loop() {
12
13   if (freq - 10 > frequency) {
14     frequency = frequency + 10 ;
15     Serial.println(frequency);
16   }
17   if (freq - 5 > frequency) {
18     frequency = frequency + 5;
19     Serial.println(frequency);
20   }
21   if (freq + 10 < frequency) {
22     frequency = frequency - 10;
23     Serial.println(frequency);
24   }
```

Picture 29. Frequency range code.

2.18 Controlled system

The proposed tuner design until this point corresponds to the following control loop:



Picture 30. Tuner control loop

The system reference would be the desired frequency, entered by the code shown in image 29, the measured error corresponds to the difference between the measured frequency and the desired frequency, the controller is the Arduino Uno, the system corresponds to the guitar and the sensor corresponds to the microphone, the amplifier, the offset and the Arduino Uno that digitize the signal, calculate its frequency and store it in the variable frequency of the algorithm.

It is a negative feedback control; the measured value is subtracted from the desired value. The design has been done in this way because the motor rotation will be in accordance with the movement of the tuner that tightens or loosens the string of the instrument.

The duration of a complete loop, from the pickup of the acoustic signal to the rotation of the tuner is 1 second. To avoid interferences between the frequency identification and the motor rotation and giving time to the user to pick the string again, has been coded a delay of 6 seconds between each complete loop.

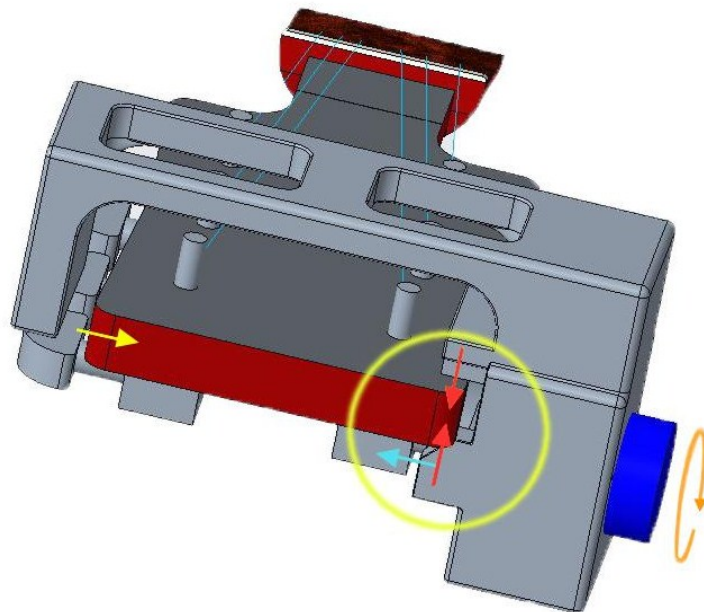
4. Structure design

In this point is explained the design of the structure, that links the guitar and the motor.

The objectives in this design are as follows:

- A design that allows fix the tuner in the guitar and restrict possible movements while tuning: It is necessary a study of forces and moments, indicated in image 31.

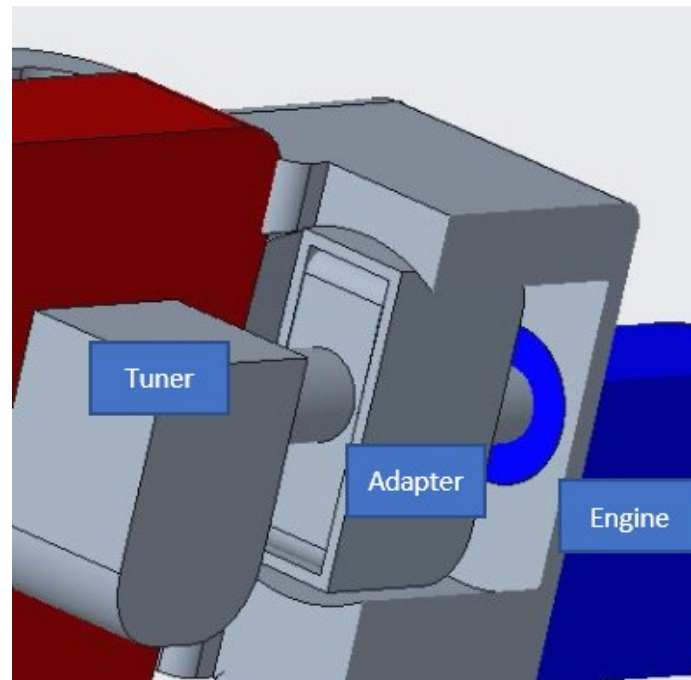
There is an external moment, indicated in the image 31, produced by the motor when rotates the tuner, in order to tune the string, this moment is compensated by some protruding parts indicated in the yellow zone, the red arrows indicate the forces of reaction. It is possible to fix the movement in the X axis, using a prisoner, the resulting force of which, is indicated with the yellow arrow, due to a diagonal protruding part we have a reaction force (light blue arrow) that balances the forces in this axis.



Picture 31. Forces and moments diagram.

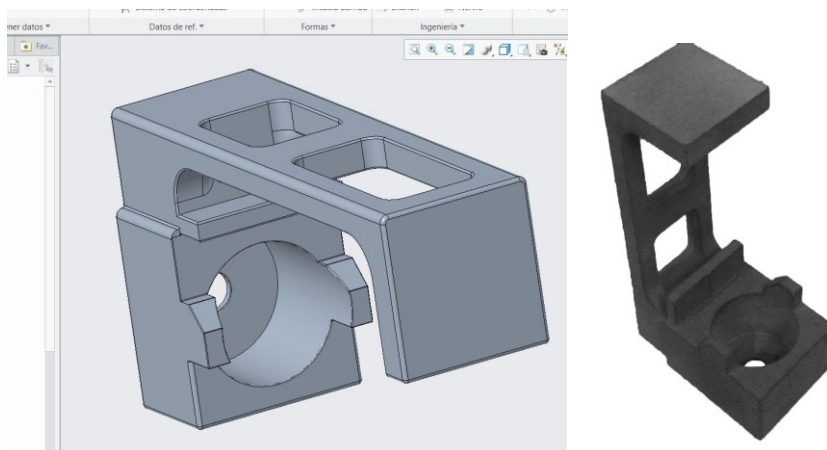
- An easy installation on the guitar: the prisoner generates a space between the headstock of the guitar and the structure that makes possible to install the tuner easily.

- The structure design must allow the motor to rotate the tuner: There is a free space inside the structure where the motor and the adapter are fixed, inside this space the motor rotate the tuner. Image 32 shows a section that reveals this space, the motor is indicated in blue.



Picture 32. Structure section

Creo parametric program is used to do a 3D design, then printed in 3D with Multifusionjet technology by HP inc.



Pictures 33 y 34 Corresponding to the 3D design and the 3D printed prototype

The following image shows the structure design installed on an Epiphone guitar.



Picture 35. Tuning structure installed on a guitar

5.Results

This section explains the tests done to the tuner and its components, to determine if the decisions taken have been right and fulfill the objectives proposed at the beginning of the project.

a. Assembly

This section explains the assembly of the structure to the guitar.

First, the motor was fixed to the structure with screws, represented by red arrows in picture 36. Then the adapter of the tuner has to be fixed to the motor axis, unfortunately the structure does not let a screw fix the adapter to the motor, so it was decided to use plastic glue.



Picture 36. Detail of motor assembly to the structure

Afterwards, a wooden piece was used to prevent damage to the guitar, due to the pressure of the prisoner.



Picture 37. Structure assembly detail

b. Motor test

Initially the stepper library was used to control the motor and was tested to make sure that the torque of the motor could rotate the tuner and check their ability to tune the guitar. In this test the motor did not apply enough torque to turn the tuner, therefore the instrument could not be tuned.

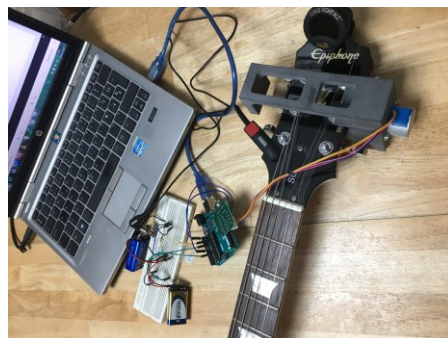
Subsequently, the power supply of the motor is considered to be the problem, the motor was tested with a 9 V supply. The motor was able to rotate the tuner when the string was not tight, but at a certain tension (lower than the tuned) did not apply enough torque to turn the tuner.

Later it was deduced, due to the shield motor, which has four LED lights which are turned on when feeding the corresponding coils, that the sequence of excitation of the coils, by the stepper library, do not let the motor achieve the maximum torque. The coil excitation sequence that prioritizes a maximum torque is manually coded and the problem was solved.

In the last test the design of the structure was tested with the motor, to verify if the structure stay fix in the guitar while tuning.

c. Guitar tuning

In this section are presented the data obtained when tuning an electric guitar, the assembly was performed for the tuning of the fifth string whose frequency is 247 Hz.



Picture 38. Prototype tuning test.

A high precision tuner was added to check the correct tuning of the instrument; the

tuner measurement is close to the desired note as shown in the following picture.



Picture 39. Test about the tuning quality of the prototype

We can conclude that the tuner fulfills the objective of tuning the string, the tuning time was 32 seconds.

To help the convergence of the algorithm, has been given a range of tolerances between the desired frequency and the captured frequency of 5 Hz, therefore the designed tuner has a maximum error of 5 Hz during tuning.

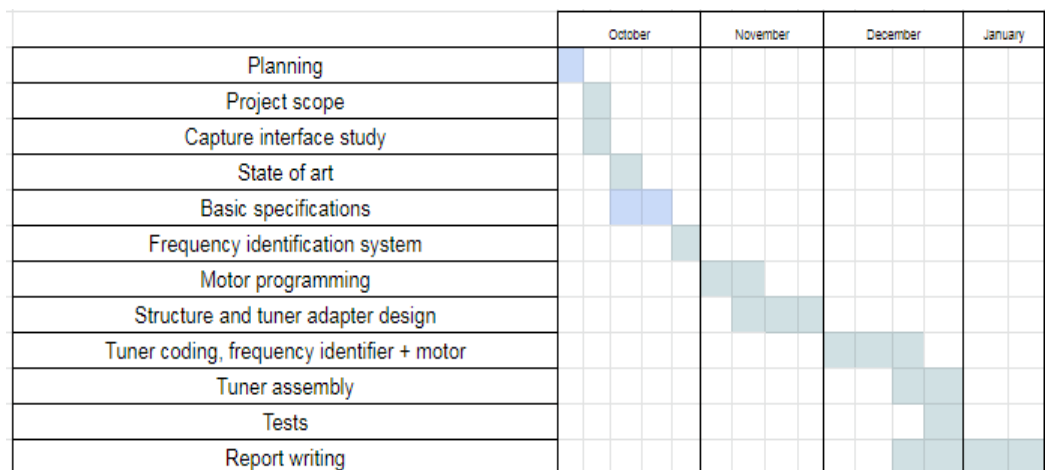
6.Planning

In order to develop a project in a period of time and complete the proposed goals, an efficient distribution of time in each of the tasks is important.

The tasks to be performed in the project are the following ones:

Planning
Project scope
Capture interface study
State of art
Basic specifications
Frequency identification system
Motor programming
Structure and tuner adapter design
Tuner coding, frequency identifier + motor
Tuner assembly
Tests
Report writing

A project planning has been done, dividing in weeks the time available, shown in the following Gantt chart.



Picture 40. Gantt chart

7. Environmental impact

The tuner is mainly composed of electronic components and a plastic structure. Electronic components are composed of several toxic elements, with the consequent impact on the environment and the risks they can cause to human health.

Among the most common substances they contain, there are elements such as cadmium, lead, lead oxide, silver, copper, antimony, nickel and mercury, among others. The recycling of these components, called electronic waste, is an expensive and polluting process, between the previous circuit and the Arduino we gather approximately 50 small components.

Since the project is focused on a DIY, “do it yourself”, the user will buy and assemble the components, for this reason, he will have a greater knowledge of the elements used and will tend to reuse the components.

8. Budget

This section indicates the costs due to the realization of this project, since the scope is limited to a prototyping phase, no other costs outside this phase will be presented.

The resources needed to carry out this project are the materials needed to build the prototype and the time spent on design, coding and construction.

Cost of the materials needed:

Item	Quantity	Unit price €	Total price €
Arduino Uno	1	8	8
Motor stepper + motor driver	1	4,5	4,5
Electronic components for the construction of the amplifier and offset circuit	1	3,4	3,4
Korg cm 200	1	7,5	7,5
Jack 3/4	1	0,4	0,4
9 V Battery	2	1	2
3D printing for the adapter	1	30	30
3D printing for the structure	1	35	35
Overall cost			90,8

Cost of engineering resources:

Item	Quantity	Unit price €	Total price €
Sound capture interface analysis.	2	15	30
Analysis of the microcontroller	3	15	45
Study of the possible methods of frequency identification by Arduino.	4	15	60
Frequency identification assembly.	8	15	120
Study of stepper motor control by Arduino.	4	15	60
Final programming, relating the measured frequency to the stepper rotation.	60	20	1.200
3D structure design.	130	20	2.600
3D design tuner adapter.	25	20	500
Complete tuner	10	15	150

assembly.			
Testing	5	15	75
Report	50	15	750
Overall cost			5.590

Total project costs: 5.680 €

9. Conclusions

➤ **The system works fine achieving initial requirements.**

The proposed concept is robust enough and deliver the required functionality according initial requirements and boundary conditions. The system is cheap and simple. This fact was demonstrated with the experiments already documented, in all the cases, the wire was properly tuned with repeatability and no differences comparing with traditional and manual tuning.

➤ **Main Learnings.**

- Selecting arduino as control system was a great decision, it is simple to program, accessible and powerful enough.
- Selecting a stepper motor was also another good decision, has simplified the motor control even without a gear box
- The system is compatible with different sounds receiver.

➤ **Industrialization. Required customization to each guitar model.**

I would like to split this topic in two aspects:

- Firmware + Electrical, that includes data acquisition + processing and motor movement management according proposed algorithm. This functionality could be common for all wires in any stringed instrument.
- Mechanical parts, that includes the structure support and the tuner adaptor. These parts must be customized to the geometry of the selected instrument.

The required customization for the mechanical parts is a weak point for this concept in case of commercialization.

The new manufacturing technologies, more appropriated for low volumes and more accessible for all of us, could help to deliver the customized parts at reasonable prizes.

➤ **Customer perception and low ratio, value added versus incremental cost and complexity.**

The proposed tuning system is automatic and doesn't require musical experience to the operator, this was demonstrated. We have some doubts about the market perception connected with this topic. Musicians used to be proud of his instruments and they doesn't used to delegate important tasks like instrument tuning to other people, they like to do it by themselves.

In addition to this I think that the extra value added (automatic and no experience

required) by this proposal, maybe is not enough compared with the extra incremental cost

➤ **Potential improvements.**

As we have said, the system works fine, most improvements are connected with the customer experience and usability:

- Add a led, to alert to the customer that a new iteration is starting, and he needs to generate a new vibration in the selected string.
- Extra volume required. The electrical PCA's could be optimized, it is obvious, but the required rigidity in the main structural body and the motor create an extra volume, that makes impossible to storage the assembly in a conventional case.
- It would be necessary to study whether the tuner could be positioned on the surface of the guitar's blade, instead of sticking out of the headstock
- Provide an external case to protect the components for external agents, such as water, dust, falls, transport vibrations, etc.
- Null aesthetic design. Th min purpose was to demonstrate the functionality and the concept, but an external case could help also in this matter.
- To save time, we could design a device that tunes all strings, one after another, but without requiring disassembling and assembling after tuning each wire. This concept could require 6 motors and a larger structural part, but the time saving is significant.

10. Bibliography

Bibliography references.

[1] Amanda ghassaei (2018). Arduino frequency detection [en línia] Instructables circuits. Disponible en: <https://www.instructables.com/id/Arduino-Frequency-Detection/>.

[2] Amanda ghassaei (2018). Arduino audio input [en línia] Instructables circuits. Disponible en: <https://www.instructables.com/id/Arduino-Audio-Input/>

Complementary bibliography

[3] Tronical Tune. [en línia] Tronical tune Plus y Disponible en: <https://www.tronicaltune.com/>

[4] Gibson guitars. [en línia] Gibson Disponible en: <https://www.gibson.com/>

[5] Arduino. Arduino control structure [en línia] Arduino. Disponible en: <https://www.arduino.cc/reference/en/language/structure/control-structure/for/>

[6] Arduino. Arduino structure [en línia] Arduino. Disponible en: <https://www.arduino.cc/reference/en/#structure>

[7] Aprendiendo Arduino. Motor paso a paso [en línia]. Disponible en: <https://aprendiendoarduino.wordpress.com/tag/motor-paso-a-paso/>

[8] Electronic components datasheet search. TL082 [en línia]. Disponible en: https://www.alldatasheet.com/view.jsp?Searchword=TI082&gclid=CjwKCAiAx_DwBRAfEiwA3vwZYqCk9xN2B-PAXmzt_uvyJahAlgtEys42WE-5yWRgCEeKebIbxEzI2hoC0PwQAvD_BwE

[9] Ester Cierco Molins: *Introducción a la acústica*. Barcelona: Septiembre 2019.

[10] Ed.xunta. Tornillo sin fin corona [en l nia]. Disponible en:
https://www.edu.xunta.es/espazoAbalar/sites/espazoAbalar/files/datos/1464947673/contido/52_tornillo_sinfncorona.html

[11] Cosas de guitarra. Gu a completa de afinadores de guitarra [en l nia]. Disponible en:
<https://cosasdeguitarra.com/afinador-de-guitarra-guia-completa/>

11. Annex

```
boolean clipping = 0;

byte newData = 0;

byte prevData = 0;

unsigned int time = 0;//keeps time and sends vales to store in timer[] occasionally

int timer[10];//storage for timing of events

int slope[10];//storage for slope of events

unsigned int totalTimer;//used to calculate period

unsigned int period;//storage for period of wave

byte index = 0;//current storage index

float frequency;//storage for frequency calculations

int maxSlope = 0;//used to calculate max slope as trigger point

int newSlope;//storage for incoming slope data

//variables for decided whether you have a match

byte noMatch = 0;//counts how many non-matches you've received to reset variables if it's been too long

byte slopeTol = 3;//slope tolerance- adjust this if you need

int timerTol = 10;//timer tolerance- adjust this if you need

//variables for amp detection

unsigned int ampTimer = 0;

byte maxAmp = 0;

byte checkMaxAmp;

byte ampThreshold = 30;//raise if you have a very noisy signal

#include <Stepper.h>

Stepper stepper(STEPS, 8, 9, 10, 11);
```

```
int freq="desired Frequency"

void setup(){

  Serial.begin(9600);

  pinMode(13,OUTPUT);//led indicator pin

  pinMode(12,OUTPUT);//output pin

  stepper.setSpeed(5);

  cli();//diabile interrupts

  //set up continuous sampling of analog pin 0 at 38.5kHz

  //clear ADCSRA and ADCSRB registers

  ADCSRA = 0;

  ADCSRB = 0;

  ADMUX |= (1 << REFS0); //set reference voltage

  ADMUX |= (1 << ADLAR); //left align the ADC value- so we can read highest 8 bits from ADCH register only

  ADCSRA |= (1 << ADPS2) | (1 << ADPS0); //set ADC clock with 32 prescaler- 16mHz/32=500kHz

  ADCSRA |= (1 << ADATE); //enable auto trigger

  ADCSRA |= (1 << ADIE); //enable interrupts when measurement complete

  ADCSRA |= (1 << ADEN); //enable ADC

  ADCSRA |= (1 << ADSC); //start ADC measurements

  sei();//enable interrupts

}

ISR(ADC_vect) //when new ADC value ready

  PORTB &= B11101111;//set pin 12 low

  prevData = newData;//store previous value

  newData = ADCH;//get value from A0
```

```
if (prevData < 127 && newData >=127){//if increasing and crossing midpoint

    newSlope = newData - prevData;//calculate slope

    if (abs(newSlope-maxSlope)<slopeTol){//if slopes are ==

        //record new data and reset time

        slope[index] = newSlope;

        timer[index] = time;

time = 0;

if (index == 0){//new max slope just reset

    PORTB |= B00010000;//set pin 12 high

    noMatch = 0;

    index++;//increment index

}

else if (abs(timer[0]-timer[index])<timerTol && abs(slope[0]-newSlope)<slopeTol){//if timer duration and slopes match

    //sum timer values

    totalTimer = 0;

    for (byte i=0;i<index;i++){

        totalTimer+=timer[i];

    }

    period = totalTimer;//set period

    //reset new zero index values to compare with

    timer[0] = timer[index];

    slope[0] = slope[index];

    index = 1;//set index to 1

    PORTB |= B00010000;//set pin 12 high

    noMatch = 0;
```

```
}

else{//crossing midpoint but not match

    index++; //increment index

    if (index > 9){

        reset();

    }

    }

    }

else if (newSlope>maxSlope){ //if new slope is much larger than max slope

    maxSlope = newSlope;

    time = 0; //reset clock

    noMatch = 0;

    index = 0; //reset index

}

else{//slope not steep enough

    noMatch++; //increment no match counter

    if (noMatch>9){

        reset();

    }

}

}

if (newData == 0 || newData == 1023){ //if clipping

    clipping = 1; //currently clipping

    Serial.println("clipping");

}
```

```
time++; //increment timer at rate of 38.5kHz

ampTimer++; //increment amplitude timer

if (abs(127-ADCH)>maxAmp){

maxAmp = abs(127-ADCH);

}

if (ampTimer==1000){

ampTimer = 0;

checkMaxAmp = maxAmp;

maxAmp = 0;

}

}

void reset(){ //clean out some variables

index = 0; //reset index

noMatch = 0; //reset match counter

maxSlope = 0; //reset slope

}

void checkClipping(){ //manage clipping indication

if (clipping){ //if currently clipping

clipping = 0;

}

}

void loop(){

checkClipping();

if (checkMaxAmp>ampThreshold){
```



```
int frequency = 38462/float(period);//calculate frequency timer rate/period

if (freq - 10 > frequency)

{

for ( int i=0 ; i<100 ; i=i+1){

digitalWrite(motorPin1, LOW);

digitalWrite(motorPin2, LOW);

digitalWrite(motorPin3, HIGH);

digitalWrite(motorPin4, HIGH);

delay(3);

digitalWrite(motorPin1, HIGH);

digitalWrite(motorPin2, LOW);

digitalWrite(motorPin3, LOW);

digitalWrite(motorPin4, HIGH);

delay(3);

digitalWrite(motorPin1, HIGH);

digitalWrite(motorPin2, HIGH);

digitalWrite(motorPin3, LOW);

digitalWrite(motorPin4, LOW);

delay(3);

digitalWrite(motorPin1, LOW);

digitalWrite(motorPin2, HIGH);

digitalWrite(motorPin3, HIGH);

digitalWrite(motorPin4, LOW);

delay(3);

}

}
```

```
delay(1000);

}

if (freq - 5 > frequency)

{

for ( int i=0 ; i<50 ; i=i+1){

digitalWrite(motorPin1, LOW);

digitalWrite(motorPin2, LOW);

digitalWrite(motorPin3, HIGH);

digitalWrite(motorPin4, HIGH);

delay(3);

        digitalWrite(motorPin1, HIGH);

        digitalWrite(motorPin2, LOW);

        digitalWrite(motorPin3, LOW);

        digitalWrite(motorPin4, HIGH);

        delay(3);

        digitalWrite(motorPin1, HIGH);

        digitalWrite(motorPin2, HIGH);

        digitalWrite(motorPin3, LOW);

        digitalWrite(motorPin4, LOW);

        delay(3);

        digitalWrite(motorPin1, LOW);

        digitalWrite(motorPin2, HIGH);

        digitalWrite(motorPin3, HIGH);

        digitalWrite(motorPin4, LOW);

        delay(3);
```

```
    }

    delay(6000);

}

if (freq + 10 < frequency) {

    for ( int i=0 ; i<100 ; i=i+1){

digitalWrite(motorPin1, LOW);

digitalWrite(motorPin2, LOW);

digitalWrite(motorPin3, HIGH);

digitalWrite(motorPin4, HIGH);

delay(3);

digitalWrite(motorPin1, LOW);

digitalWrite(motorPin2, HIGH);

digitalWrite(motorPin3, HIGH);

digitalWrite(motorPin4, LOW);

delay(3);

digitalWrite(motorPin1, HIGH);

digitalWrite(motorPin2, HIGH);

digitalWrite(motorPin3, LOW);

digitalWrite(motorPin4, LOW);

delay(3);

digitalWrite(motorPin1, HIGH);

digitalWrite(motorPin2, LOW);

digitalWrite(motorPin3, LOW);

digitalWrite(motorPin4, HIGH);
```

```
    delay(3);

}

delay(1000);

}

if (freq + 5 < frequency) {

    for ( int i=0 ; i<50 ; i=i+1){

        digitalWrite(motorPin1, LOW);

        digitalWrite(motorPin2, LOW);

        digitalWrite(motorPin3, HIGH);

        digitalWrite(motorPin4, HIGH);

        delay(3);

        digitalWrite(motorPin1, LOW);

        digitalWrite(motorPin2, HIGH);

        digitalWrite(motorPin3, HIGH);

        digitalWrite(motorPin4, LOW);

        delay(3);

        digitalWrite(motorPin1, HIGH);

        digitalWrite(motorPin2, HIGH);

        digitalWrite(motorPin3, LOW);

        digitalWrite(motorPin4, LOW);

        delay(3);

        digitalWrite(motorPin1, HIGH);

        digitalWrite(motorPin2, LOW);

        digitalWrite(motorPin3, LOW);

        digitalWrite(motorPin4, HIGH);
```

```
        delay(3);  
    }  
    delay(6000);  
    }  
    }  
    delay(100);  
  
}
```