**ECS 472 Digital Signal Processing Term Project**

**Automatic Guitar Tuner**

**Chatrin Phunruangsakao 5922783501**  
**Poblarb Sakdasukon 5922790480**  
**Wachiraporn Suriyatanakul 5922791140**  
**Junwen Zhang 5922800354**  
**Mohamed Mahdi Benes 6122048208**

**Introduction**

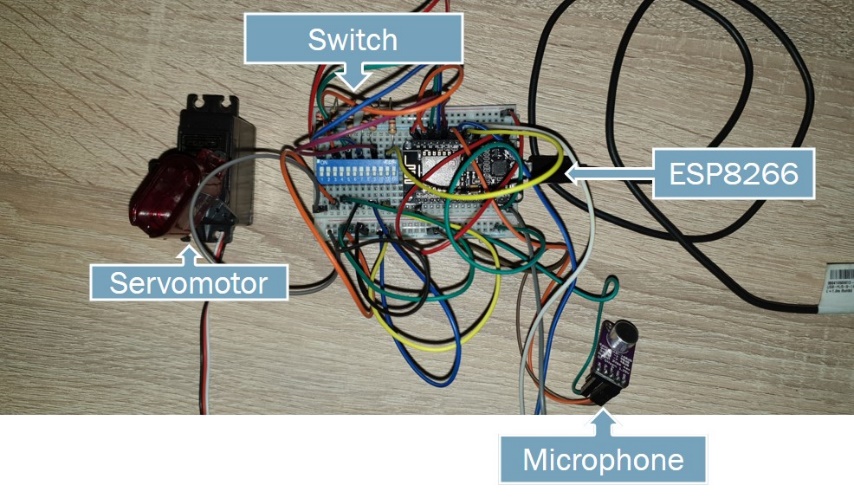
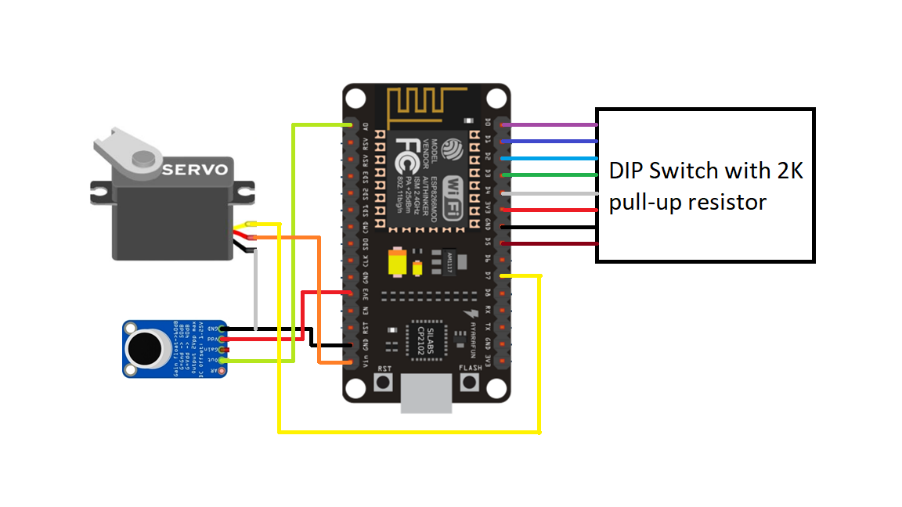
This project focuses on the design and construction of automatic guitar tuner. The guitar tuner functions by receiving out-of-tune guitar string sound, produced by manually plucking the string, and compare it with the reference sound, embedded in the microcontroller, µC, along with necessary code for tuning, to make suitable adjustment. Moreover, the user may indicate the desired string to tune by turning on and off the mechanical switches. The prototype circuitry is illustrated in Figure 1.

Figure 1: Prototype Circuitry of Automatic Guitar Tuner

**Apparatus**

* Microcontroller (ESP8266)
* DIP switch
* Microphone Module (MAX9814)
* DC Servo Motor with gripper
* 6 2KΩ Resistors
* Jumper Wires
* Protoboard

**Implementation**

1. **System Design**

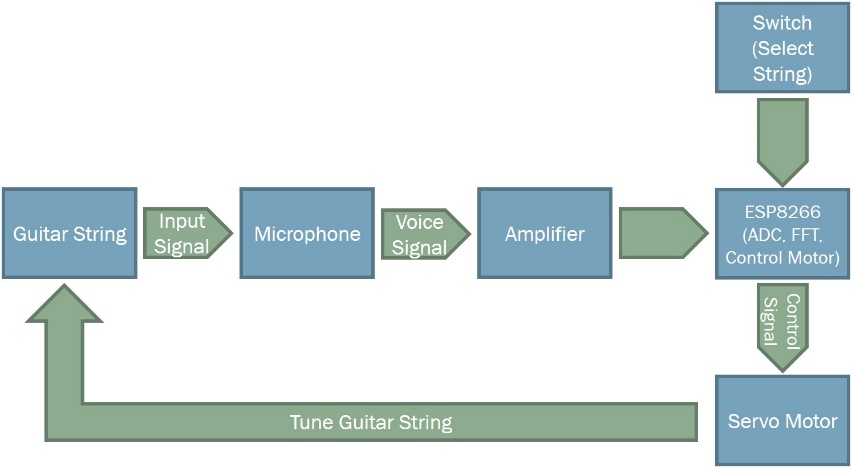
The block diagram depicted in Figure 2 describes the system design in a feedback loop. The DIP switch acts like an interface for user to pick which string to tune. Guitar string sound is fed into µC module where the sound signal is transformed to electrical signal and later the electrical signal is amplified. The amplified electrical signal is then sent to µC, ESP8266. There, the signal undergoes *Analog-to-Digital Conversion (ADC)* and *Fast Fourier Transform (FFT)* to find *Major Peak Frequency*, which is used to compared with the reference frequency of the tuning string for the µC to decide to whether to tune up, tune down or stop tuning. Subsequently, when the µC has made its decision, it controls DC servo motor tuning routine.

Figure 2: System Block Diagram

|  |  |
| --- | --- |
| String | Reference Frequency (Hz) |
| EL | 82.41 |
| A | 110.00 |
| D | 146.83 |
| G | 196.00 |
| B | 246.94 |
| EH | 329.63 |

Table 1: Guitar String Frequencies for 440Hz Notes

1. **Algorithm**

In this repository, https://github.com/pchatrin/AutomaticGuitarTuner, both Arduino code for µC, and MATLAB code to verify that the tuned guitar string by automatic tuner is actually in tune, are provided. However, this section only discusses the fundamental algorithm of Arduino code, demonstrated in Figure 3. The µC has two inputs, namely, string to tune and string sound. By inputting string to tune, the µC automatically selects the reference or target frequency to tune to. The string sound is fed into µC and from there it goes through ADC and FFT to find peak frequency. µC later calculates the frequency error or margin from the reference frequency and uses it to decide whether to command servo motor to tune in which direction. If the frequency error is less than 1 Hz then servo motor is told to tune up, vice versa, if the frequency is more than 1Hz then servo motor is told to tune down. The loop continues until the frequency is within 1Hz frequency error margin. Thereafter, the user may begin to tune the next string by repeating the steps.

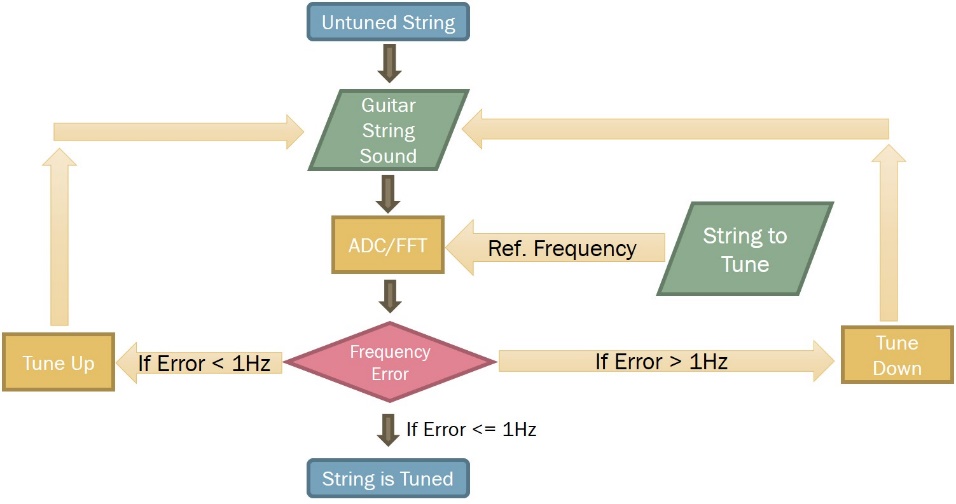
The sampling frequency is set at 1000 samples per second due to the fact that at this sampling frequency the original signal with frequency less than or equal to 500Hz can be reconstructed without aliasing and guitar strings frequency ranges from 82.41 to 329.63Hz. Although out-of-tune string frequency may vary from this value, normally the frequency does not rise up to 350Hz, therefore sampling at 1000 Sa/s is adequate enough. Furthermore, to fully utilize the capability of µC, the number of samples used for FFT are fully allocated in µC’s memory is each cycle, thus enhancing accuracy of the system.

Figure 3: Logic Flow of the System

**Result**

The tuning results are compared with commercial mobile guitar tuner application *Guitar Tuna* and MATLAB code, shown in Table 2. Guitar Tuna is an application that informs the user that the string is out-of-tune by displaying plus (+), to indicate that the current string frequency is too high, and minus (-), to indicate that the current string frequency is too low, along with a number to indicate the margin from reference frequency.

The first column of each trials is the result obtained from Guitar Tuna application, while the second column of each trials is the result obtained by performing FFT in MATLAB to string sound tuned by automatic tuner. The MATLAB code gives the string frequency in Hertz.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| String | Trial 1 | | Trial2 | | Trial 3 | |
| EL | 0 | 82.2265 | +3 | 83.1213 | +1 | 82.2584 |
| A | 0 | 109.7578 | +2 | 110.8126 | +3 | 111.3442 |
| D | +4 | 149.5732 | +2 | 147.2246 | -2 | 145.6199 |
| G | -2 | 194.5267 | -2 | 194.9130 | -2 | 194.1855 |
| B | +2 | 247.9674 | +0 | 246.9438 | +1 | 247.6254 |
| EH | -2 | 326.0324 | -2 | 327.9997 | -3 | 325.7813 |

Table 2: Result from Tuning

**Conclusion**

This project is aiming to automatically adjust the guitar tunes. By selecting the string to tune with the help of the switch, the automatic guitar tuner detects the tune guitar string through the microphone, and it treats the sound using reference frequencies in order to adjust the tune with the help of a servo motor. The Guitar Tuna application and a MATLAB code are used to check that the guitar tune is not more out-of-tune. However, the accuracy of the tuner can be improved by employing filters, more sampled data, and higher precision motor.

Figure 4: QR code for YouTube Video Explaining System Design and Demonstration