*Instrument tuner*

Product Design Specification

Designer: *Andrew Timmons, Brian Cummings*

Version: *1.0*

*04/21/2023*

TABLE OF CONTENTS

[1 Introduction 3](#_Toc133421024)

[1.1 Purpose of The Product 3](#_Toc133421025)

[2 General Overview and Design Guidelines/Approach 3](#_Toc133421026)

[2.1 General Product Overview 3](#_Toc133421027)

[2.2 Assumptions / Constraints / Standards 3](#_Toc133421028)

[3 Architecture Design 4](#_Toc133421029)

[3.1 Hardware Architecture 4](#_Toc133421030)

[3.2 Software Architecture 4](#_Toc133421031)

[3.3 Performance Considerations 5](#_Toc133421032)

[3.4 Power Considerations 6](#_Toc133421033)

[3.5 Sensors/Actuators Description 6](#_Toc133421034)

[4 System Design 6](#_Toc133421035)

[4.1 Use-Cases 6](#_Toc133421036)

[4.2 Data Conversions 6](#_Toc133421037)

[4.3 Application Program Interfaces 6](#_Toc133421038)

[4.4 User Interface Design 7](#_Toc133421039)

[4.5 Performance 7](#_Toc133421040)

[4.6 Bill of material (BOM) 7](#_Toc133421041)

[4.7 Calibration and test procedures 7](#_Toc133421042)

[5 Conclusion on Product Design Specification 8](#_Toc133421043)

[Appendix A: References 9](#_Toc133421044)

[Appendix B: Key Terms 10](#_Toc133421045)

# Introduction

## Purpose of The Product

This product can be used to tune a stringed instrument to a specific tuning. Each string can be tuned to a target frequency by following LED indicators that show whether the string’s frequency is too low, too high, or on pitch. This provides a simple way for musicians to tune their instrument without needing to be able to recognize the target note by ear.

# General Overview and Design Guidelines/Approach

## General Product Overview

*This product is designed for the user to tune each string of their instruments one at a time. The user first plays a single string with the microphone close enough to pick up a sufficient audio signal from the instrument. The signal is sampled by the microcontroller, and the approximate frequency of the note is calculated. The frequency in the pre-selected tuning array nearest to the calculated frequency is selected as the target frequency, and the string corresponding to this note is indicated using the LED bar. The LED bar uses a single LED for each string, with the highest string on the right and the lowest string on the left. Finally, the on-board RGB LEDs on the microcontroller are used to indicate whether the calculated frequency is higher than (blue LED), lower than (green LED), or within the frequency tolerance range of the target frequency (red LED). The process of sampling the audio, calculating the frequency, and displaying relevant tuning information repeats on a loop, allowing the user to continually make tuning adjustments until they have achieved the proper note.*

## Assumptions / Constraints / Standards

For the current implementation of the product, our first assumption is that the instrument being tuned is a six-stringed guitar being tuned to standard tuning. However, more tunings with different numbers of notes can be added in the future, which nullify this assumption.

The next assumption is that the user can place the microphone sufficiently close to the instrument in order for the sensor to pick up an adequately strong audio signal. A signal that is too weak may give the wrong result or even produce no result at all. Similarly, we assume that the environment is reasonably noise-free; if the sensor picks up too much ambient noise, the frequency may be calculated incorrectly.

Finally, a frequency tolerance range of +/- 1 Hz has been used for determining whether a calculated frequency matches its target frequency. A 1 Hz discrepancy may still be detectable to a trained ear, and a smaller tolerance around +/- 0.5 Hz may be a more ideal choice for accuracy. The current tolerance range is a compromise for performance, as a greater resolution would require a faster sampling rate and a greater number of audio samples. It is also a compromise for the cheap microphone being used, which may struggle to differentiate finer changes in frequency.

# Architecture Design

## Hardware Architecture

The microphone detects the sound and captures it as a voltage waveform, amplifies this signal, and outputs an analog signal routed to the microcontroller’s OPAMP1 input. OPAMP1 applies a 2x amplification to the analog signal, and its output is routed to the microcontroller’s ADC1, which is set to continuous mode. The microcontroller timer TIM2 is used to generate timer interrupts in which ADC1 samples are collected and stored in software, thereby controlling the ADC sampling rate, which is set to sample 1024 samples at 4 kHz. When all samples are collected, the digital signal is processed in software, and LEDs on the LED bar are set via GPIOE 7 – 12 to indicate the target note and the onboard RGB LEDs are set via GPiOA9, GPIOB7, and GPIOC7 to indicate the relation between the target and calculated frequencies. Figure 1 provides a high-level hardware diagram outlining this process.

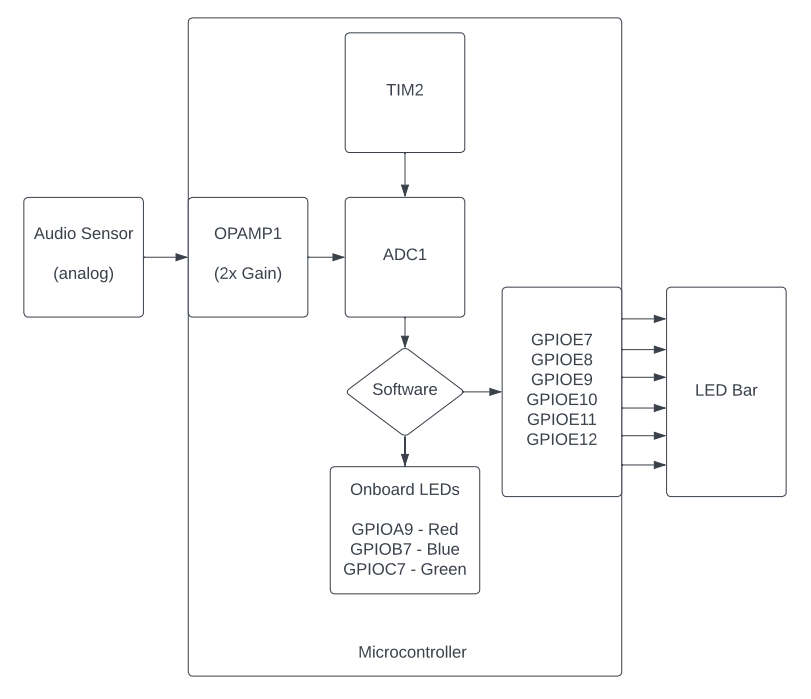


Figure : High Level Hardware Diagram

## Software Architecture

The product’s software architecture consists of three phases: the sampling gathering phase, the digital signal processing phase, and the LED setting phase. These three phases happen one after another, then repeat within a while loop for as long as the program is running.

The getSample function is the primary function of the sample gathering phase. This function simply resets timer TIM2, resets the sample iterator global variable which tracks how many samples have been gathered, enables ADC1 conversion, and then enables TIM2, which will begin to trigger timer interrupts. Each timer interrupt reads a single conversion result from the ADC1 data register, stores it in the next available slot in the sample array, and increments the sample iterator. The getSample function blocks the program from advancing using a while loop that waits until the number of samples gathered via timer interrupt reaches 1024. At this point, TIM2 is disabled, ADC1 conversion is disabled, and the sample array bears a new digital signal to process.

The purpose of the digital signal processing phase is to calculate the fundamental frequency of the digital audio signal gathered in the previous phase. This phase uses the fft and hps functions imported from the signal\_processing file. The fft function performs an in-place fast Fourier transform on the sample array, such that the contents of the sample array are transformed into the digital signal’s frequency domain representation. The FT array is then passed into the hps function, which performs the Harmonic Product Spectrum algorithm in order to determine the fundamental frequency from the FT. However, we found that only considering one or two harmonics in the FT spectrum yielded the best results in terms of performance and accuracy, which causes the algorithm to simply detect the highest magnitude in the FT and output its index. The frequency of the audio signal is then obtained from this index using the formula

(1)

The final phase first uses the calculated frequency from the previous phase to select the nearest frequency in the tuning array as the target frequency via the getNearestNote function from the instrument\_tuner file. The setLEDBar function indicates the target note on the LED bar using the index of the target note in the tuning array. Finally, the calculated and target frequencies are passed into the setBoardLEDs function, which calculates the difference between the two and sets the onboard RGB LEDs based on the result.

Figure 2 shows a high level software diagram for the repeating main loop of the program. Function calls are indicated in diamonds, while input/output data is indicated with squares.

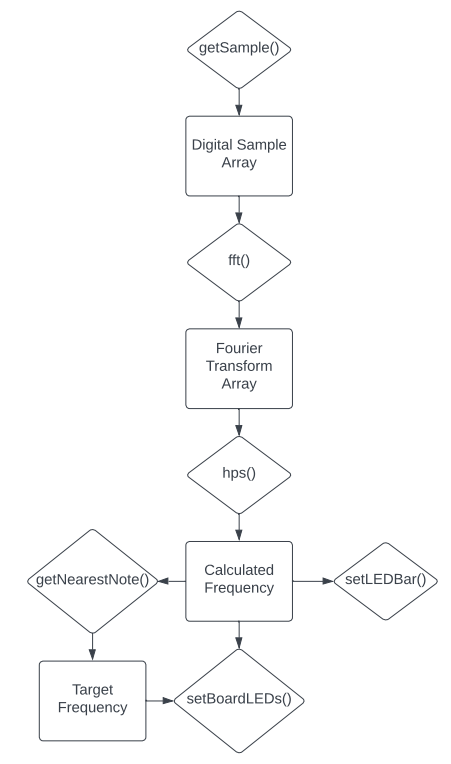


Figure : High Level Software Diagram

## Performance Considerations

Performance considerations for this product are primarily focused around the signal processing algorithms used on the digital audio signal. In particular, the fast Fourier transform algorithm requires a significant amount of processing power that takes some time to execute. This becomes problematic if the delay between the user playing the note on their instrument and the LED tuning indicators lighting up becomes too great, as this will make it difficult to make real-time adjustments. To help reduce the time needed to execute the main program loop, the following steps were taken:

1. Setting the microcontroller system clock to 110 MHz, it’s maximum clock speed, which directly speeds up all of the processing done by the microcontroller.
2. Using only the first half of the FT spectrum in the harmonic power spectrum to halve the time the HPS algorithm takes to determine the fundamental frequency. This takes advantage of the fact that the Fourier transform is symmetrical along the x-axis, so only the first half is truly needed.
3. Choosing a sampling frequency and number of samples collected that gives accurate frequency calculations without overwhelming the FFT algorithm with too many samples. This was determined experimentally, starting with finding an acceptable number of samples to take, then choosing a sampling frequency that was fast enough to capture the waveform in enough detail without being so fast that an insufficient time duration of the waveform was captured, which could negatively affect FFT accuracy.

With these measures in place, the product performs well; it is accurate while also being fast enough to display tuning information in a reasonable time after playing a note.

## POWER CONSIDERATIONS

The KY-037 sound sensor requires 5V to operate. This program places a much greater priority on performance than power by setting the system clock to its maximum speed, so this product may be relatively power hungry as a result.

## SENSORS/aCTUATORS DESCRIPTION

This sensor emits a signal if the microphone of the sensor detects a noise. The sensitivity of the sensor can be adjusted by means of a controller.

Digital output: Via the potentiometer, a limit value for the received sound can be set, at which the digital output should switch.

Analog output: Direct microphone signal as voltage level

LED1 : Shows that the sensor is powered

LED2 : Indicates that a noise has been detected

FUNCTION OF THE SENSOR

This sensor has three functional components on its circuit board: The front sensor unit, which physically measures the environment and outputs it as an analog signal to the second unit, the amplifier. This amplifies the signal depending on the resistance set on the rotary potentiometer and sends it to the analog output of the module.

Here it is to be noted: The signal is inverted. If a high value is measured, this results in a lower voltage value at the analog output.

The third unit represents a comparator, which switches the digital output and the LED when the signal falls below a certain value. This value (and thus the sensitivity of the module) can be adjusted via the rotary potentiometer.

# System Design

## Bill of material (BOM)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Part Name | Part Number | Per Unit | Total Quantity Required | Unit Cost |
| [STM32 Microcontroller](https://estore.st.com/en/products/evaluation-tools/product-evaluation-tools/mcu-mpu-eval-tools/stm32-mcu-mpu-eval-tools/stm32-nucleo-boards/nucleo-l552ze-q.html) | NUCLEO-L562ZE-Q | 1 | 1 | $20.85 |
| [sound sensor](https://campaign.aliexpress.com/wow/gcp/tesla-pc-new/index?UTABTest=aliabtest377151_530968&src=google&src=google&albch=shopping&acnt=708-803-3821&slnk=&plac=&mtctp=&albbt=Google_7_shopping&albagn=888888&isSmbAutoCall=false&needSmbHouyi=false&albcp=19131229154&albag=&trgt=&crea=en2251832528996846&netw=x&device=c&albpg=&albpd=en2251832528996846&aff_fcid=7b8626e2b5a74daaa28589943a7d44bc-1682299304150-07178-UneMJZVf&aff_fsk=UneMJZVf&aff_platform=aaf&sk=UneMJZVf&aff_trace_key=7b8626e2b5a74daaa28589943a7d44bc-1682299304150-07178-UneMJZVf&terminal_id=d51c1082168946a39c05ba30e636096d&wh_weex=true&wx_navbar_hidden=true&wx_navbar_transparent=true&ignoreNavigationBar=true&wx_statusbar_hidden=true&bt_src=ppc_direct_lp&scenario=pcBridgePPC&productId=2251832528996846&OLP=1085100208_f_group2&o_s_id=1085100208) | KY-037 | 1 | 1 | $0.29 |
| [Jumper Wires](https://www.amazon.com/Elegoo-EL-CP-004-Multicolored-Breadboard-arduino/dp/B01EV70C78/ref=sr_1_2_sspa?keywords=jumper%2Bwires&qid=1682297804&s=industrial&sprefix=jumper%2Bwir%2Cindustrial%2C172&sr=1-2-spons&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUExMlNPU0U2TzRKNjlaJmVuY3J5cHRlZElkPUEwOTQ1MDgyMVpYVDUwQlU5UzJKSSZlbmNyeXB0ZWRBZElkPUEwOTQ1MzIxMUVLUFZPSjk1OTFYOSZ3aWRnZXROYW1lPXNwX2F0ZiZhY3Rpb249Y2xpY2tSZWRpcmVjdCZkb05vdExvZ0NsaWNrPXRydWU&th=1) | - | ~5 | 5 | $6.98 |

Needs LED bar, bread board, bar resistor, smaller jumper wire pack probably

## Calibration and test procedures

Calibration of the sensor potentiometer.

# Conclusion on

[Summarize your experience with this project. What challenges you faced, did you mean the specifications, any ways to improve]

Appendix A: References

[Insert the name, version number, description, and physical location of any documents referenced in this document. Add rows to the table as necessary.]

The following table summarizes the documents referenced in this document.

|  |  |  |
| --- | --- | --- |
| **Document Name and Version** | **Description** | **Location** |
| *<Document Name and Version Number>* | *[Provide description of the document]* | *<URL or Network path where document is located>* |

Appendix B: Key Terms

*[Insert terms and definitions used in this document. Add rows to the table as necessary. Follow the link below to for definitions of project management terms and acronyms used in this and other documents.*

*http://www2.cdc.gov/cdcup/library/other/help.htm*

The following table provides definitions for terms relevant to this document.

|  |  |
| --- | --- |
| **Term** | **Definition** |
| *[Insert Term]* | *[Provide definition of the term used in this document.]* |
| *[Insert Term]* | *[Provide definition of the term used in this document.]* |
| *[Insert Term]* | *[Provide definition of the term used in this document.]* |