*Instrument tuner*

Product Design Specification

Designer: *Andrew Timmons, Brian Cummings*

Version: *1.0*

*04/29/2023*

TABLE OF CONTENTS

[1 Introduction 3](#_Toc133673124)

[1.1 Purpose of The Product 3](#_Toc133673125)

[2 General Overview and Design Guidelines/Approach 3](#_Toc133673126)

[2.1 General Product Overview 3](#_Toc133673127)

[2.2 Assumptions / Constraints / Standards 3](#_Toc133673128)

[3 Architecture Design 4](#_Toc133673129)

[3.1 Hardware Architecture 4](#_Toc133673130)

[3.2 Software Architecture 5](#_Toc133673131)

[3.3 Performance Considerations 6](#_Toc133673132)

[3.4 POWER CONSIDERATIONS 7](#_Toc133673133)

[3.5 SENSORS/aCTUATORS DESCRIPTION 7](#_Toc133673134)

[4 System Design 8](#_Toc133673135)

[4.1 Bill of material (BOM) 8](#_Toc133673136)

[4.2 Calibration and test procedures 8](#_Toc133673137)

[Conclusion on Product Design Specification 9](#_Toc133673138)

[Appendix A: References 10](#_Toc133673139)

[Appendix B: Key Terms 11](#_Toc133673140)

# 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.

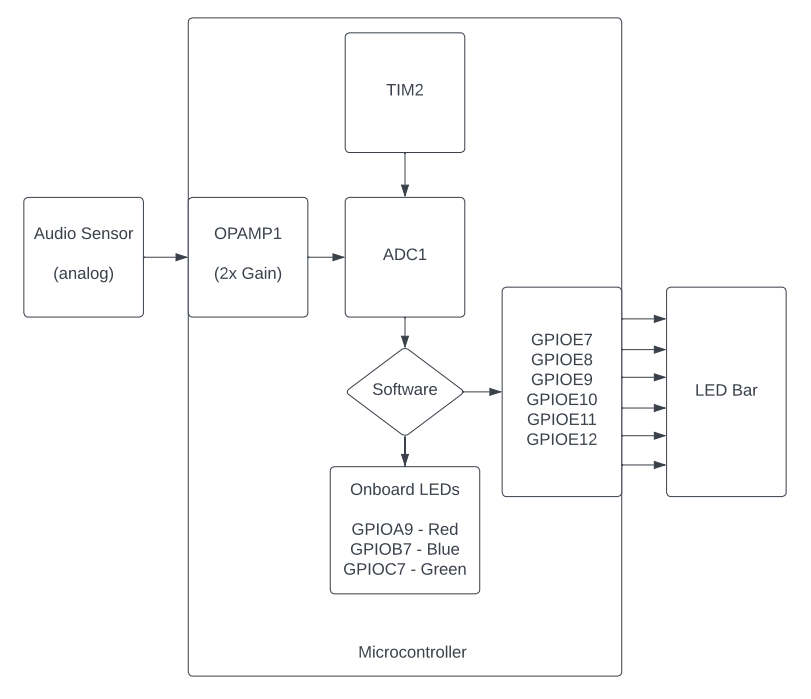


Figure : High-Level Hardware Diagram

## 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.

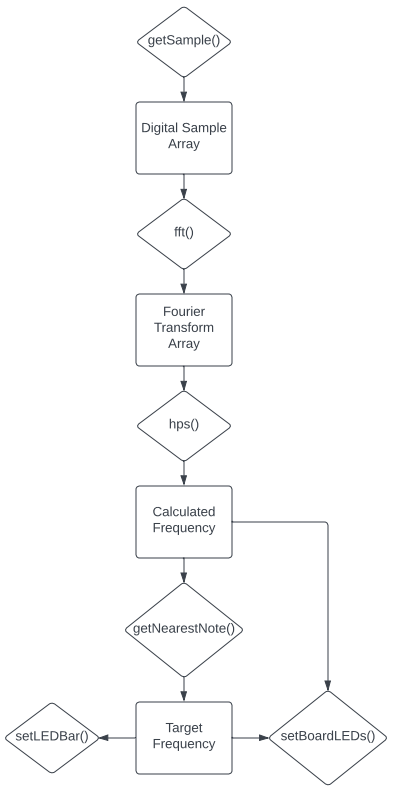


Figure : High-Level Software Diagram

## 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, its 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

The KY-037 sensor continuously detects sound and emits an analog signal that captures the sound’s waveform. This sensor has three functional components on its circuit board: a front sensor unit responsible for detecting sound and converting it to an analog voltage, an opamp and an adjustable potentiometer for amplifying the signal, and a comparator used for switching the digital output pin (not used in this application). The potentiometer adjusts the sensor’s sensitivity/amplification and common mode voltage (the resting voltage when no sound is detected). Note that the output analog signal is actually an inversion of the original audio signal.

The sensor has two onboard LEDs:

* LED1: Shows that the sensor is powered.
* LED2: Used to indicate the ideal setting for the potentiometer. When set correctly, the LED will blink when a noise is detected.

Because our product further amplifies the analog signal using the microcontroller’s opamp, the sensor’s “ideal setting” as indicated by LED2 can be ignored.

# System Design

## Bill of material (BOM)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Part Name | Supplier | 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) | ST Microelectronics | 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) | AliExpress | KY-037 | 1 | 1 | $0.29 |
| [Jumper Wires](https://www.temu.com/40pcs-dupont-line-3-93in-7-87in-male-to-female-jumper-wire-dupont-cable-for-arduino-diy-g-601099515547017.html?top_gallery_url=https%3A%2F%2Fimg.kwcdn.com%2Fproduct%2FFancyalgo%2FVirtualModelMatting%2F02283aa492d07c329ce9398906a1108b.jpg&spec_gallery_id=19999306&refer_page_el_sn=209279&_x_ads_channel=google&_x_ads_sub_channel=shopping&_x_login_type=Google&_x_vst_scene=adg&_x_ns_sku_id=17592191846143&_x_gmc_account=647900107&_x_ads_account=5532219654&_x_ads_set=19987815521&_x_ads_id=146789406903&_x_ads_creative_id=655335349433&_x_ns_source=g&_x_ns_gclid=%7Bgclid%7D&_x_ns_placement=&_x_ns_match_type=&_x_ns_ad_position=&_x_ns_product_id=17592191846143&_x_ns_target=&_x_ns_devicemodel=&_x_ns_wbraid=%7Bwbraid%7D&_x_ns_gbraid=%7Bgbraid%7D&_x_ns_targetid=pla-297194214843&refer_page_name=kuiper&refer_page_id=13554_1682643725132_om1itil18h&refer_page_sn=13554&_x_sessn_id=ddj98zhv4g&is_back=1) | Mi Miao Miao c digital | - | 40 | 5 | $1.99 |
| [Resistor Array](https://www.digikey.com/en/products/detail/bourns-inc/4610X-101-102LF/1089168) | Digi-key | 4610X-101-102LF | 1 | 1 | $0.55 |
| [LED Light Bar](https://www.digikey.com/en/products/detail/liteon/LTA-1000HR/153273) | Digi-key | LTA-1000HR | 1 | 1 | $1.38 |
| [Breadboard](https://www.digikey.com/en/products/detail/dfrobot/FIT0096/7597069) | Digi-key | FIT0096 | 1 | 1 | $2.90 |
| Total |  |  |  |  | 27.96 |

## Calibration and test procedures

The KY-037 sound sensor’s potentiometer must be adjusted before use in order for the device to capture and output a usable analog signal. Ideally, the microcontroller’s ADC should output a digital value around 3000 DN out of 4096 DN when no sound is present, which would normally translate to a sensor output voltage of approximately 2.42V. However, because the input voltage to the microcontroller is amplified by 2x, the correct resting output voltage from the sensor should be approximately 1.21V. Connect the sensor’s power and ground so that the sensor is being powered, connect the analog output to a multimeter, and adjust the potentiometer until a voltage of 1.21V is achieved. This should be done under relatively quiet conditions.

To verify that the sensor is working, power the sensor and adjust the potentiometer until LED2 is just on the threshold of turning off. At this potentiometer setting, any sufficiently loud sound should cause LED2 to blink, indicating that an audio signal is being detected. The analog output of the sensor can also be tested using an oscilloscope set to trigger slightly above or below the resting voltage.

The performance/accuracy of the instrument tuner can be verified either using a different tuner or by ear. Once the instrument has been tuned using the instrument tuner, a different tuner should indicate the expected note for each string, and there should be no dissonance when playing specific chords on your instrument at the same time as a recording of that same chord.

# Conclusion on

Overall, the instrument tuner works as expected to an acceptable degree of performance and accuracy. Now that the product is complete, the architecture and logic of the product is relatively simple and straightforward. The most difficult parts of this project were working with the KY-037 sound sensor to acquire a usable analog signal and calculating the fundamental frequency of that signal. To overcome this challenge, the sensor was connected to an oscilloscope and a tone with a known frequency was played, which confirmed that the microphone was sensing and outputting an appropriate signal, but the amplitude of this signal was far too small to be used practically. Amplifying the signal using the microcontroller’s opamp solved this issue, though the FFT and HPS algorithms were still not outputting correct data. To fix this, the microcontroller’s LPUART was used to display both the digital audio signal as sampled by the ADC and the digital FT of that signal. These signals allowed us to adjust the sensor’s potentiometer to an ideal setting that uses as much of the microcontroller’s dynamic range as possible. It also revealed that a large spike at index zero of the digital FT array – likely a result of the DC contributions of the analog signal – was causing the signal processing to fail. After accounting for this, the signal processing gave relatively reliable results, which were further refined by experimentally adjusting the sampling rate and number of samples being taken.

While the product is functional in its current implementation, there exists room for future improvement. As mentioned before, the current implementation only supports one instrument with one tuning, but more can be added in software. More tuning choices would also necessitate an easy way to select each choice via some kind of user interface. For example, a seven-segment display and a push button could be used to cycle through tunings until one is selected. As a bonus, the seven-segment display could then be used to display the target note and the calculated note to the user. The product could further be improved using a better signal processing algorithm. Most battery-powered guitar tuners don’t use the Fourier transform – instead they use an algorithm that counts the number of “zero crossings” of the time-domain analog signal and estimate the frequency. While not as accurate, this process is much faster which can make up for the loss of accuracy. As this algorithm requires less processing, it could also allow for a reduction in clock speed, which would improve the product’s power performance. Finally, a better sound sensor would further improve the accuracy of the product. Though cheap, the KY-037 is better suited for noise detection rather than for capturing detailed audio waveforms. Better still, an accelerometer that attaches to the instrument could be used instead of a sound sensor. An accelerometer would have the advantage of reducing the impact of ambient noise pollution, although it would limit the instruments that could be tuned with this product.

Appendix A: References

The following table summarizes the documents referenced in this document.

|  |  |  |
| --- | --- | --- |
| **Document Name and Version** | **Description** | **Location** |
| *KY-037 Microphone sensor module (high sensitivity)* | *Technical data/ description of the KY-037 sound sensor* | *https://electropeak.com/learn/download/ky-037-datasheet/#* |
| STM32L552xx and STM32L562xx advanced Arm®-based  32-bit MCUs | *Reference manual for the NUCLEO-L562ZE-Q microcontroller* | *https://www.st.com/resource/en/reference\_manual/dm00346336-stm32l552xx-and-stm32l562xx-advanced-arm-based-32-bit-mcus-stmicroelectronics.pdf* |
| *LED Display Product Data Sheet LTA-1000HR* | *Technical data sheet of the LTA-1000HR LED light bar* | *https://optoelectronics.liteon.com/upload/download/DS-30-92-0810/LTA-1000HR.pdf* |
| *4600X Series - Thick Film Conformal SIPs* | *Technical data sheet of the 4610X-101-102LF resistor array* | *https://www.bourns.com/docs/Product-Datasheets/4600x.pdf* |
| *400 Tie Point Interlocking Solderless Breadboard* | *Technical data sheet for the FIT0096 bread board* | *https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/FIT0096\_Web.pdf* |

*This product uses an FFT algorithm found from the following online resource:*

<https://rosettacode.org/wiki/Fast_Fourier_transform>

Appendix B: Key Terms

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

|  |  |
| --- | --- |
| **Term** | **Definition** |
| *ADC* | *Analog to Digital Converter* |
| *FFT* | *Fast Fourier Transform* |
| *FT* | *Fourier Transform* |
| *GPIO* | *General Purpose Input/Output* |
| *HPS* | *Harmonic Product Spectrum* |
| *LPUART* | *Low Power Universal Asynchronous Receiver / Transmitter* |
| *TIM* | *Timer* |