# Auto Tuning Drum Key

Final Report
CEN4908-C
Spring 2025

Brandon Davis
Ismael Maura
Marc Perlas
Komlan Tchoukou
Ramses Ziane-Cherif

*Sound Sync.*
University of Florida
Computer Engineering Dept.

# Table of Contents





# 1 Abstract

This project presents the development and implementation of an automatic drum tuning device designed to assist drummers in achieving precise and consistent tuning across all drum lugs. The system integrates hardware and software components, including an ESP32 microcontroller, an I2S microphone, a continuously rotating motor, and a companion mobile application, to provide an efficient and user-friendly drum tuning experience.

The device captures the frequency of drum strikes using the I2S microphone, processes these signals through Fast Fourier Transform (FFT) algorithms executed on the ESP32, and determines the required adjustments for each drum lug. Bluetooth connectivity facilitates communication between the ESP32 and the mobile application, enabling users to specify the number of lugs on the drum and the desired tuning note. Upon initiation, the device automatically adjusts each lug, tightening or loosening as necessary, until the measured frequency matches the target pitch within an acceptable tolerance.

The mobile application offers an intuitive interface, providing clear visual guidance throughout the tuning process. Key features enhancing usability include real-time frequency visualization, structured navigation, and built-in safeguards to prevent incorrect tuning sequences. Comprehensive testing confirmed the device's ability to deliver accurate and reliable drum tuning performance, resulting in a cohesive, fully integrated solution.



# 2 Introduction

Learning how to play a new instrument is a unique challenge. Learning proper handling, correct playing form, and even routine maintenance are all parts which can prove challenging, moreso for those picking up an instrument on their own.

Maintenance especially is something that is often overlooked, as it is not as kinetic or exciting as learning to play a note for the first time, or playing a rhythm exercise. However, maintenance is still a major part in learning an instrument, and if neglected, will only present problems later down the line. One of the most noticeable parts of the maintenance process is tuning.

Drums, specifically snare drums, present a unique challenge in learning to tune for the first time. Other instruments, compared to the snare drum, have relatively simple tuning processes. As an example, with the Saxophone, one needs to place the mouthpiece more in for a sharper tuning, or more out for a flatter tuning. The Trombone follows a similar process, except instead of a mouth piece, one moves the tuning slide located near the rear of the instrument.

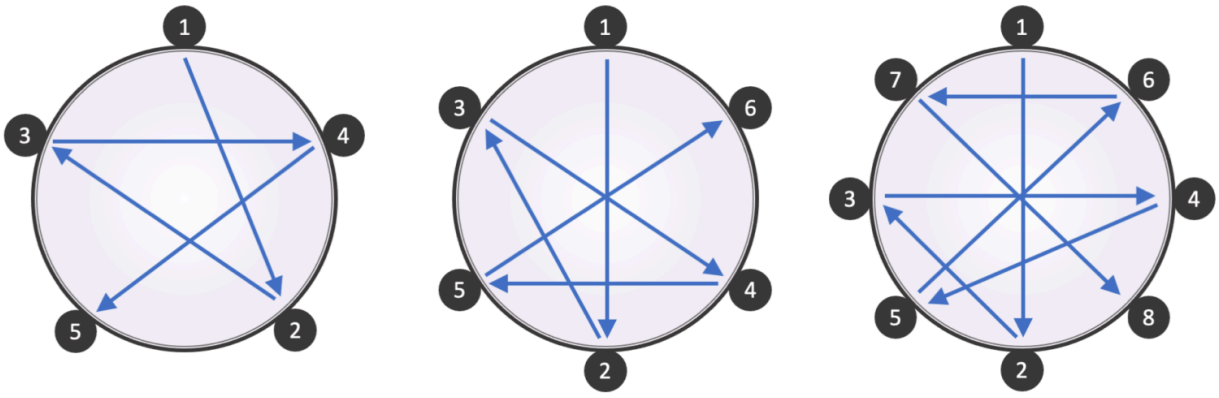


*Figure 1: The pattern must follow the lugs in to ensure proper tuning [1]*

The tuning process in a snare drum is more involved. As seen in the figure above, the process is arduous, requiring a key to tune each individual section of the drum, and this needs to be done in a star pattern. While tuning a drum does not require the same amount of frequency as a saxophone or a trombone, the process is time consuming, strenuous, and can cause confusion for people learning how to play the instrument for the first time.

It is for this reason that the Auto Tuning Drum Key can serve as an assistant not only to experienced percussionists, but especially to beginners who wish to pick up the snare and want to learn an important aspect of playing the snare. The Auto Tuning Drum Key is not only a device to help tune a drum, but through the use of the interface application is also one which can teach a beginner to the snare drum how to tune a drum. The main goal of this project is to help tune to allow beginners to easily learn this vital process so they can get onto other aspects of learning the drums itself.

# 3 Background

## 3.1 Theoretical Background

The main algorithm used in this project is the Fast Fourier Transform (FFT), which converts time-domain data into the frequency domain, breaking down complex sounds into a combination of frequencies. While the Discrete Fourier Transform (DFT) is slow to compute, the FFT speeds this up from $O(n^2)$ to $O(n \log n)$. FFT calculations are handled by the ArduinoFFT library on the ESP32.

Additionally, the Harmonic Product Spectrum (HPS) technique is used to estimate the fundamental frequency of harmonic signals, such as musical instruments. HPS compresses the frequency spectrum by integer factors (e.g., 2, 3, 4), amplifying the fundamental frequency while suppressing noise and non-harmonic components. This makes HPS a reliable method for pitch detection.

Other tuners, like the Auto Guitar Tuner developed by students at the Institut Informatika Indonesia, also use FFT for accurate frequency detection and tuning

## 3.2 Prior Art

Auto-Tuning tools are nothing new to the market of musical instrument accessories. There exist auto-tuning tools, but this is mostly in the realm of guitars. Most other instruments do not have such auto-tuning devices, perhaps due to the way these instruments tune being not as extensive. Perhaps the most prolific of these is the Tronicaltune Guitar Tuner, as seen in the figure below. The device itself is a kit, which requires the end-user to take out the initial lugs on the head of the guitar to bold in the device to automatically tune the guitar. They have kits for both 6-line guitars, where all lugs are arranged in a horizontal line, and 3+3 guitars, where the lugs are placed vertically, in a 3 and 3 arrangement.

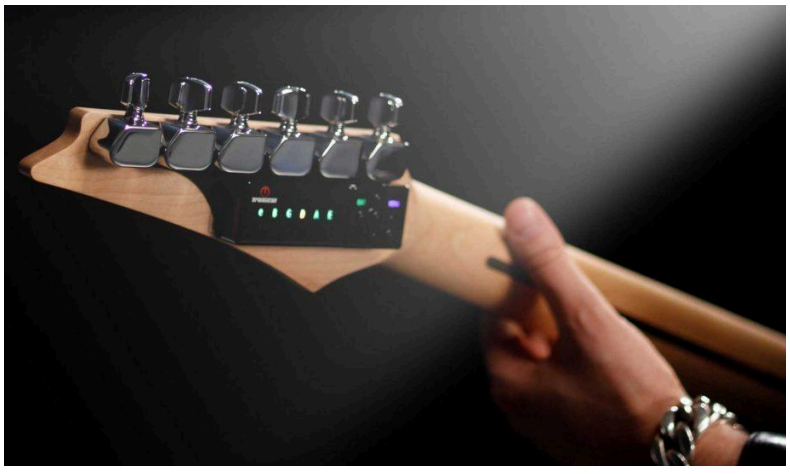

*Figure 3: The Tronicaltune Guitar Tuner as seen a 6-line guitar [3]*

One criticism of these devices lies in the fact that they require this extensive set-up. Removing the lugs on a guitar head is not an easy task, and is one that is prone to error if done incorrectly. In addition, this device can only be used on the guitar that it gets attached to. What this means is that using the auto tuner on another guitar requires the removal of the device from one guitar to repeat the process on another.

The Auto-Tuning Drum Key differs in that the device itself is simply the key that works in conjunction with an application to tune the drum. While the Tronicaltune Guitar Tuner does not require an app, the Auto-Tuning Drum Key can work with most snare drums and does not require an extensive set-up. The Auto-Tuning Drum Key is also not a standalone device, requiring an application in order for the device to work. Asides from that, the key can be placed on most snare drum lugs, and all the user needs to do is select the correct lug count, connect to the device via bluetooth, and then play each individual section of the drum for the key to turn in the right direction.

## 3.3 Experimentation

During the design of the Auto Tuning Drum Key, we have found several factors that can greatly frequency readings of the played drum. This can range from environmental factors, such as the atmosphere around when played and the surface on which the drum is being played, but also the material of the drum. For the purposes of our testing, we have used the same drum. We later learned that without muting the bottom of the drum itself, the reading for the section of the drum would not be as accurate. This would later be remedied by using a blanket.

Testing for the drum key was done after the encasing was done. This would be done to further refine the algorithm of signal processing using the FFT. Overall, this would make us need to loosen the boundaries of our algorithms on both the application and the firmware side.



# 4 Timeline

## 4.1 Completed External Interface Work

- Finalized encasing
  - The encasing has been finalized with white filament for better improved user intuitiveness in regards to power supply, charging state, and open mic status.
  - Clearance holes and screws have been placed to keep everything secure and robust alone with fasteners to keep both halves of the encasing together further improving build quality
  - Research on GDT methods, prior Solidworks experience, and assistance from mechanical engineering peers assisted in completion
- Circuitry verification
  - Components have been integrated via the perfboard with completed soldered circuitry between the ESP, mic, power supply, and motor
  - Completion was done via schematic creation and prior soldering experience
- Torque verification
  - Constant testing has been done to make sure the motor has enough torque to tighten a lug. Without sufficient torque tuning would be impossible
  - Research on typical kg/cm^3 for a lug was done in comparison to those available to us. Along with sufficiently strong material and adhesive were used to make sure the motor can be used at full capacity

## 4.2 Completed Internal Interface Work

- Bluetooth transmission
  - A stable Bluetooth connection enables communication between the app and the ESP32. The app sends the target frequency, while the ESP32 transmits the current note. This is done using Android Studio Kotlin's requestBluetoothPermissionLauncher, requestBluetoothSocketPermissionLauncher, bluetoothManager, and bluetoothAdapter. Data is transmitted via bluetoothSocket using input and output streams, with SerialBT.println for sending and SerialBT.readStringUntil for receiving data.
- Error mapping
  - The frequency discrepancy is mapped to the motor's turn radius, allowing for faster tuning. Larger rotations close the gap, while smaller rotations fine-tune the frequency. The turn angle is calculated by multiplying the frequency difference by 5, with a constraint between -180 and 180 degrees. If the angle exceeds 3 degrees, the motor rotates within these bounds.
- Open mic
  - A switch controls communication between the ESP32 and the app, preventing the mic from picking up unwanted frequencies. Once the target note is reached, the mic is automatically turned off to lock in the frequency. The app sends a valid target frequency when the switch is on and 0 when off. If the target frequency is 0 or less, the motor stops.
- UX/UI Color
  - The app's color spectrum on the current note TextView and buttons provides a visual indicator of progress toward the target. This helps users, especially non-musicians, intuitively track tuning. The color is calculated based on the difference between the current and target

notes, with green indicating closeness to the target and red indicating the difference. Once the threshold is reached, the color turns green.

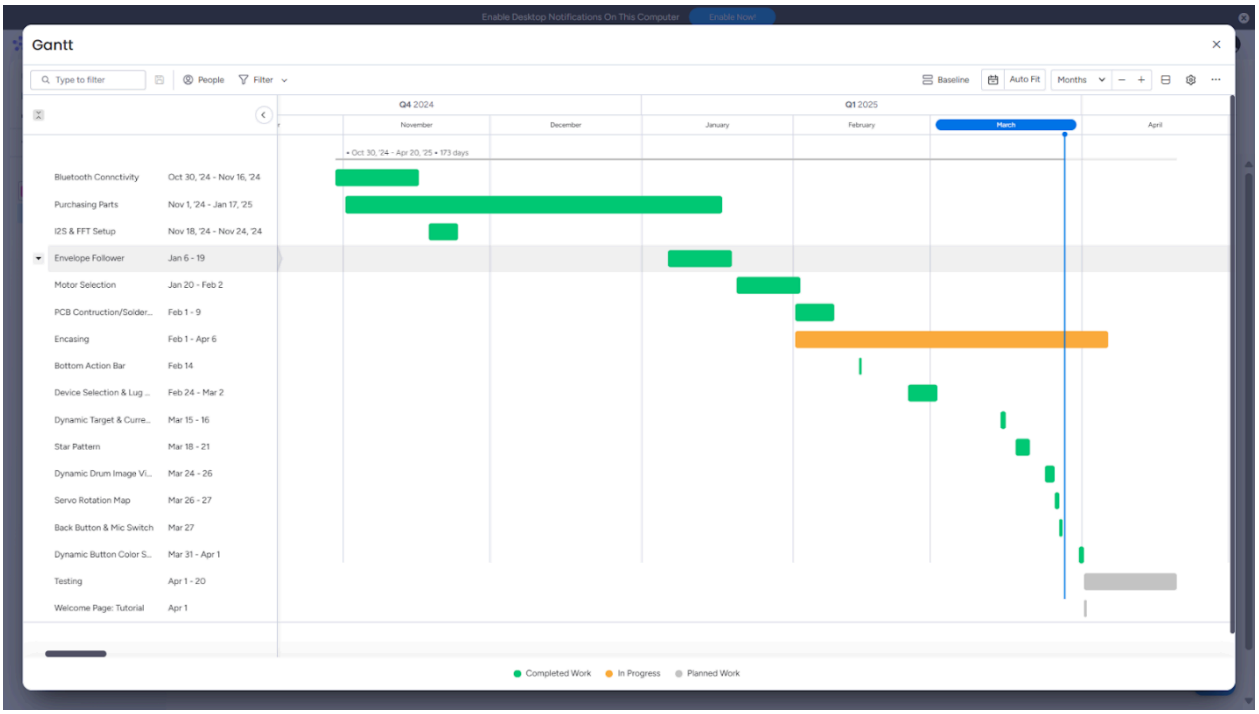


*Figure 5: Auto Tuning Drum Key Gantt Graph*

# 5 Design

## 5.1 External Interface

### 5.1.1 User Interaction

User interaction occurs through the mobile app, requiring minimal hardware involvement—just turning on the device and holding it during tuning. Users select the drum's lug count, set the desired note, and record each drum strike's frequency. The device tunes automatically as the user continues striking the drum. The app includes a triangular guide for striking specific areas and highlighted buttons for the tuning sequence. The ESP32's blue LED turns off when a lug is tuned, signaling the user to move to the next lug in a star pattern.

### 5.1.2 App Interface & Controls

The app prioritizes lug count selection and real-time note detection. The interface shows the current lug's running status, turning from red to green when the target frequency is reached. The current note box displays the proximity to the target note, with color changes indicating tuning accuracy. Users select the tuning device from a dropdown menu and follow the star pattern tuning process, with a tutorial for first-time users.

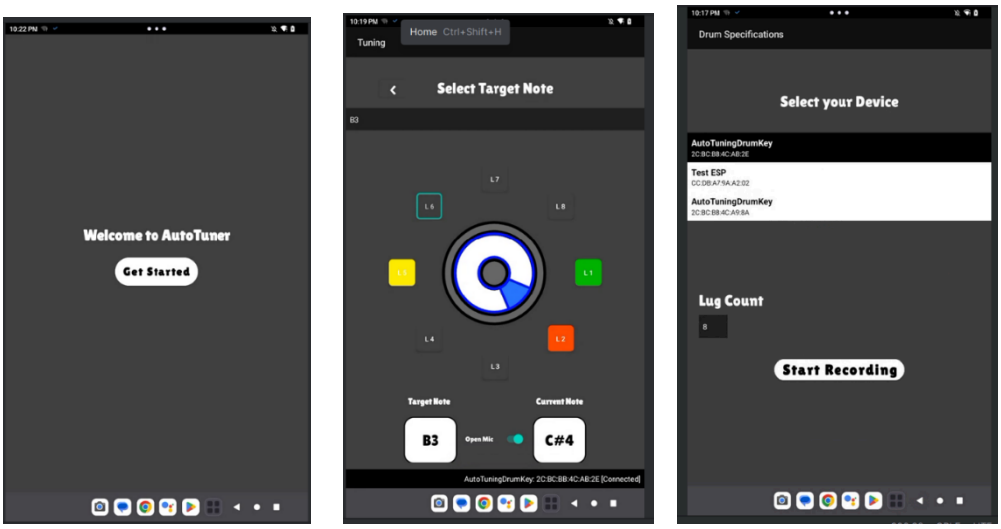


*Figure 6: UI for the Companion App*

### 5.1.3 Hardware Interface

The ESP32 microcontroller, microphone, and motor are compactly connected on a perf board. The switch and ports are repositioned for accessibility, ensuring easy charging and operation. The encasing conceals internal components, protects from damage, and features a power switch and a charging port. The motor operates with sufficient torque (13 kg*cm) to ensure effective lug adjustments via a drill bit that is adhesively stuck to a motor attachment, and the attachment itself is also connected adhesively to the motor.



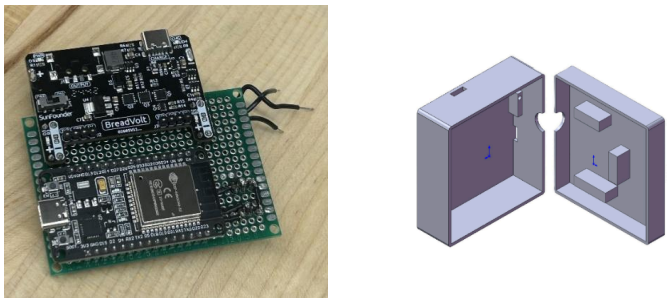

*Figure 7: On the left is the perf. board, the middle is the solidworks model, and the right is the 3d Printed final model used in the project.*

## 5.2 Communication Protocol

The system relies on classic Bluetooth for communication. The backend will store user preferences like the lug count and ESP32 mac-address. The tuning process remains unchanged, with real-time note detection and adjustments via microphone I2S communication and motor PWM.

## 5.3 Persistent State

### 5.3.1 Software Persistent State

The app provides UI feedback, reads Bluetooth frequency data, and sends target frequency commands. Currently, the app supports 25 different note frequencies that can be sent to the ESP32 to tune the drum. The Classic Bluetooth communication model ensures reliable data transfer.

### 5.3.2 Hardware Persistent State

The ESP32 processes microphone data via I2S and uses FFT for frequency extraction, with a threshold-based system for accurate data collection. A moving average filter smooths the microphone samples, and the harmonic product spectrum (HPS) estimates the fundamental frequency, useful for detecting pitch in drums with harmonic overtones.

The motor adjusts based on whether the detected frequency is within ±10 Hz of the target, making finer adjustments as the frequency nears the target. The ESP32 manages FFT sampling memory in the heap. The enclosure design ensures portability while securely housing the components.

## 5.4 Internal State

### 5.4.1 Component Architecture

The system consists of:

- **Hardware:** ESP32 microcontroller, I2S microphone, LED indicator, FFT processing, harmonic product spectrum, moving average filter, servo motor, and a battery pack.



- **Software:** Android app for user interaction and Bluetooth communication.

The ESP32 receives a target frequency from the app, processes microphone input, and adjusts the motor accordingly. The app displays real-time data to guide the user. The encasing also helps provide a stable environment for the tuning process.

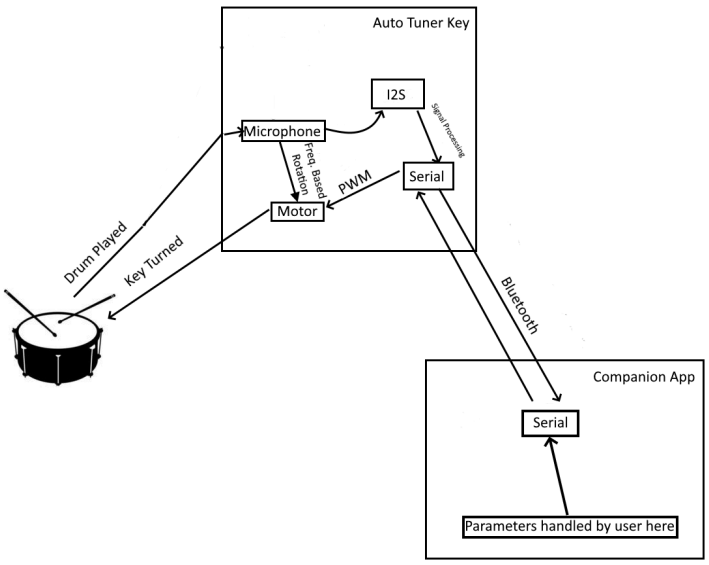


*Figure 8: A control flow diagram for how the component architectures interact.*

### 5.4.2 Hardware Components

- **ESP32 Microcontroller:** Manages Bluetooth, audio processing, and servo control.
- **I2S Microphone:** Captures and digitizes drum sounds.
- **Servo Motor:** Applies torque for lug adjustments.
- **Battery Pack:** Delivers appropriate voltages to all components.
- **Encasing:** Provides a stable environment for tuning and protects internal circuitry.

### 5.4.3 Software Components

- **ESP32 Firmware:** Controls microphone input, LED indicator, FFT processing, harmonic product spectrum, moving average filter, motor logic, and Bluetooth communication.
- **Android Application:** Handles device pairing, displays note accuracy, and manages the full tuning workflow, and gives color UI indicators for note accuracy.

### 5.4.4 Algorithms

- **FFT (Fast Fourier Transform)**: Core for pitch detection.

- **Harmonic Product Spectrum (HPS)**: Improves pitch recognition in overtone-heavy audio.
- **Envelope Follower**: Peak detection for cleaner signal analysis.
- **PID Loop**: Ensures stable and responsive tuning behavior.

## 5.5 Risks and Mitigation

Electric tuners are often seen as tools for beginners, but they may be viewed as a crutch that prevents learning essential techniques like tuning by ear. The Auto Tune Drum Key could face similar criticism for removing the need to tune by ear and use a drum key, potentially diminishing a key skill in learning an instrument.

Another concern is the risk of drum damage if the Auto Tuner fails to recognize tension limits or attempts to tune to an unattainable frequency. Caution is needed to ensure the Auto Tuning Drum Key doesn't attempt to tune to an impossible frequency, which could damage the drum.



# 6 Tools & Standards

## 6.1 Firmware Tools

- FFT; <arduinoFFT.h>
- I2S; <driver/i2s.h?
- PWM; <ESP32Servo.h>
- Bluetooth; “BluetoothSerial.h”

## 6.2 Software Tools

- Bluetooth; android.bluetooth.BluetoothAdapter
- Broadcast; android.content.BroadcastReceiver
- Context; android.content.Context
- Intent; android.content.Intent
- IntentFilter; android.graphics.Color
- Bundle; android.os.Bundle
- Log; android.util.Log
- View; android.view.View
- ViewTreeObserver; android.view.ViewTreeObserver
- AppCompactActivity; android.appcompat.app.AppCompatActivity
- SwitchCompat; android.appcompat.widget.SwitchCompat
- IOException; java.io.IOException
- cos; kotlin.math.cos
- sin; kotlin.math.sin
- AnimatorSet; android.animation.AnimatorSet
- ObjectAnimator; android.animation.ObjectAnimator
- DialogInterface; android.content.DialogInterface
- ColorStateList; android.view.animation.LinearInterpolator
- AlertDialog; androidx.appcompat.app.AlertDialog

## 6.3 Hardware Tools

- 3D Printing
- Power Supply; BreadVolt
- Microcontroller; ESP32
- Mic; INMP441
- Motor; TD-8135MG
- PCB; Perfboard

## 6.4 Engineering Standards

### 6.4.1 Process

- Version control was crucial for managing contributions to both hardware and software, ensuring smooth integration of features.



- The firmware followed the Arduino Style Guide, prioritizing readability, with comments used to explain code and keep team members informed. A protocol of commenting rather than deleting was used to improve code rather than replacing it.
- Extensive testing was conducted with different drums, frequency ranges, and firmware modifications to determine the most efficient tuning method. Online tuning apps and tone generators were used for frequency verification and controlled notes during development.

### 6.4.2 Protocols

- Bluetooth communication between the ESP32 and the app uses SPP via BluetoothSerial.h and the BluetoothAdapter API, ensuring a stable connection with RFCOMM standards for handling timeouts and reconnection.
- I2S audio streaming follows the Philips standard, configured with ESPIDF's driver/i2s.h library at 44.1kHz, 16-bit sample size, and a 1024 buffer length.
- PWM controls the servo using ESP32Servo.h following the RC PWM standard.
- FFT processing via arduinoFFT.h converts raw audio into average frequencies, with an envelope follower smoothing spikes for easier peak detection.
- Broadcast receivers and intents handle app-to-system interactions, using IntentFilter and BroadcastReceiver in compliance with Android's broadcast lifecycle and permissions model.

## 6.5 Design Constraints

- A key design constraint is that different drums fall within different frequency ranges, making the device suitable for tom drums but less effective for snare drums.
- The tuning range is limited by the drill bit material. While the servo motor and lug are metallic, the drill bit is plastic, restricting the amount of torque the device can apply based on the drill bit's strength and its grip on the servo motor's teeth.



# 7 Impact Analysis

## 7.1 Needs

The automatic drum tuner provides a reliable, efficient, and user-friendly solution for precise drum tuning. It addresses key needs:

**Accuracy:**

Achieves precise tuning (±5 Hz) using a PID control loop and real-time frequency detection.

**Ease of Use:**

Simplifies the process with an intuitive mobile app interface and automated adjustments.

**Instrument Safety:**

Protects drumheads and lugs with safety limits (±90-degree motor range, torque halt).

**Consistency:**

Delivers repeatable results, reducing variability from manual tuning.

**Portability:**

A compact, ergonomic device for use in various settings, such as studios and live performances.

## 7.2 Impact

The automatic drum tuner enhances the tuning experience for both amateur and professional musicians with several key benefits:

**Improved Efficiency:**

Motor adjustments and PID control reduce tuning time, enabling quicker setups.

**Enhanced Precision:**

Refined microphone threshold and real-time feedback ensure accurate pitch detection.

**User Empowerment:**

The intuitive app and visual feedback make tuning accessible to non-technical users.

**Instrument Longevity:**

Safety features prevent over-tightening, extending the lifespan of drumheads and lugs.

**Reliability:**



Stable Bluetooth connectivity and persistent state management reduce user frustration.

**Market Potential:**

Its unique automation, portability, and affordability could disrupt traditional tuning tools.

## 7.3 Limitations and Drawbacks

Despite its strengths, the system has several limitations that could impact its effectiveness and adoption:

**Drum-Specific Calibration:**

The microphone threshold and motor adjustments are optimized for a single drum type, potentially requiring recalibration for other drum sizes, materials, or brands, which could limit versatility.

**Manufacturing Precision:**

The 3D-printed enclosure suffers from small tolerances due to printer resolution.

**Lack of Persistent Storage:**

Without a backend database, user preferences (e.g., lug-specific notes) are not saved across sessions, limiting personalization and convenience.

**Bluetooth Dependency:**

The system relies entirely on Bluetooth, which may face interference in noisy environments, and lacks alternative connectivity options (e.g., Wi-Fi).

## 7.4 Future Work

To address limitations and enhance the system's value, the following improvements are proposed:

**Multi-Drum Compatibility:**

Develop adaptive algorithms for microphone and motor calibration to support various drum types, increasing market reach.

**Enclosure Refinement:**

Invest in higher-resolution manufacturing (e.g., injection molding) to improve enclosure tolerances, PCB alignment, and support removability, enhancing durability and aesthetics.

**App Enhancements:**

Expand on robust error handling to prevent crashes.



Add a backend database or a cloud solution to save user preferences persistently.

**Connectivity Options:**

Explore Wi-Fi connectivity as an alternative to Bluetooth, improving reliability in varied environments.

**Haptic Feedback:**

Add subtle motor vibrations or app-based haptic cues to signal tuning completion, enhancing user interaction.

Testing and Validation: Conduct extensive testing across different drums, environments, and user skill levels to ensure reliability and gather feedback for iterative improvements.

UI Polish: Further refine the app's visual feedback (e.g., animations for lug transitions) and enclosure aesthetics to align with professional-grade products.

## 7.5 Ethical and Professional Considerations

The development and deployment of the automatic drum tuner raise several ethical and professional considerations:

**User Safety and Trust:**

The system must reliably prevent damage to instruments, as promised by safety limits. Any failure could erode user trust.

Transparent communication about limitations (e.g., single-drum calibration) is essential to avoid misleading users about the system's capabilities.

**Accessibility:**

The app's minimal interface benefits users with varying technical skills, but future iterations should consider accommodations for visually or hearing-impaired users (e.g., high-contrast modes, auditory cues).

**Data Privacy:**

Although no backend exists yet, future implementations with persistent storage must prioritize secure handling of user preferences to comply with data protection laws.

Bluetooth communication should be encrypted to prevent unauthorized access to frequency **data or device control.**

**Professional Integrity:**



Collaboration with drum manufacturers or professional drummers for endorsements should be transparent and avoid conflicts of interest.

# 8 Results

## 8.1 Tests

During the alpha and beta milestones, extensive testing was done on the drum tuner system's critical components. Initial tests focused on the envelope follower's performance during drum strikes in a controlled lab environment. The goal was to ensure the envelope quickly responded to amplitude spikes and filtered out background noise. The envelope rose sharply with each strike and decayed smoothly, capturing the necessary sound for pitch analysis. Threshold sensitivity was later refined to reduce false positives and extend the sampling window for better pitch detection. Below is a picture of the drum strikes:

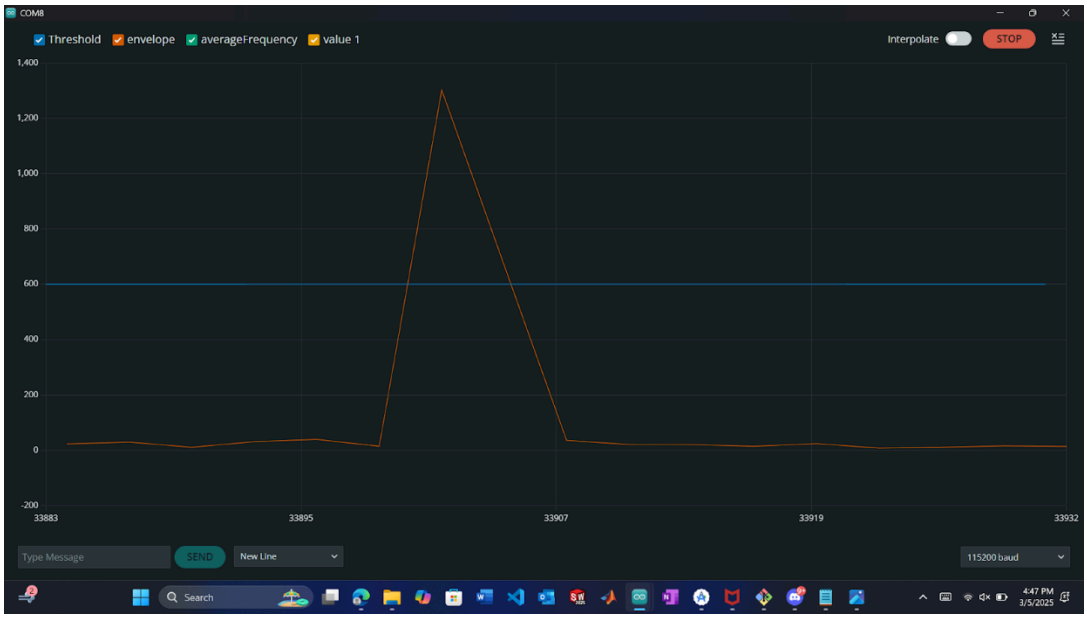


*Figure 9: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc id mi urna. Quisque suscipit et diam a aliquet. Aliquam sed iaculis massa, sollicitudin vehicula justo.*

This image shows envelope follower testing during drum strikes. The envelope smooths spikes above the threshold, demonstrating the effectiveness of the recordAndCalculate function, which captures sound before resonance fades.

To improve pitch stability, a moving average filter was tested with window sizes of 3, 5, 7, and 9 during real-time tuning using both live hits and synthetic tones on the ESP32. A 5–7 point window balanced responsiveness and noise reduction best, leading to the adoption of a 5-point filter in the final pipeline.



The Harmonic Product Spectrum (HPS) algorithm was also tested for improved pitch detection in low/mid frequencies, where fundamentals are often masked. Tests using snare and floor toms in a dampened room showed HPS outperformed peak detection in consistency and accuracy, leading to its inclusion in the final design.

Motor control was validated by detuning and re-tuning drums across a wide range. The ESP32-controlled motor, using a linear PID-like loop, made accurate, gradual adjustments as the target pitch was approached. Testing confirmed a torque requirement of 12.5 kg·cm, prompting an upgrade to a higher-torque motor and finer control logic near the target.

```
23:32:31.196 -> Current Freq: 788.10 | Target Freq: 250.00 | Error: -538 | Angle: -90
23:32:46.466 -> averageFrequency:278.62
23:32:47.668 -> Current Freq: 278.62 | Target Freq: 250.00 | Error: -28 | Angle: -28
23:32:55.498 -> averageFrequency:262.51
23:32:56.629 -> Current Freq: 262.51 | Target Freq: 250.00 | Error: -12 | Angle: -12
23:33:13.968 -> averageFrequency:255.48
23:33:15.114 -> Current Freq: 255.48 | Target Freq: 250.00 | Error: -5 | Angle: -5
23:33:33.996 -> averageFrequency:250.39
23:33:33.996 -> Adjustment too small - holding position
```

*Figure 10: Linear controller testing for the correct pitch modularly changing*

The mechanical enclosure and drill adapter underwent rigorous testing. Five 3D-printed enclosure versions were evaluated for port alignment, screw hole accuracy, and structural integrity. Fitment tests revealed issues with standoff hole alignment and power switch support, which were fixed in later versions. The drill adapter initially failed under torque, prompting reinforcement of the motor coupling and use of adhesive to prevent slippage.

Bluetooth and app functionality were tested through simulated pairings and user workflows, including frequency selection, tuning, and screen navigation in lab and rehearsal settings. A disconnection bug triggered by the back button was resolved. Real-time feedback was improved with a color-coded interface (green for in-tune, red for out-of-tune), using frequency-based interpolation. The UI was also updated to require frequency selection before tuning.

Power and battery life were verified during full tuning sessions. The battery maintained stable voltage and current throughout, requiring no system changes.

## 8.2 Final State

The final state functions as intended. Users can select their device and drum specs before tuning. Key features—including open mic, star pattern guidance, color mapping, and fine-tuning control—operate correctly. The open mic ensures frequency detection only occurs when the user initiates it or wants to analyze specific drum sections.



Color scaling applies to both TextViews and buttons: TextViews use broader thresholds to show frequency proximity, while buttons use precise scaling for accurate tuning. Highlighted buttons guide users through a star pattern, promoting balanced lug tuning by alternating across the drum.

Fine-tuning control maps the frequency discrepancy to servo rotation, with larger angles for bigger differences and smaller ones for fine adjustments, ensuring accurate pitch alignment within the defined threshold.


*Figure 11: This shows the how the average frequency is linerized based on the envelope waveform.*

Testing for these aspects were done through constant run through of total drum tuning. This gave us many repetitions to see what worked and what didn't. These repetitions demonstrated the color scaling and rotation mapping. Open mic verification was shown through whether or not serial plotter data was present depending on its state.

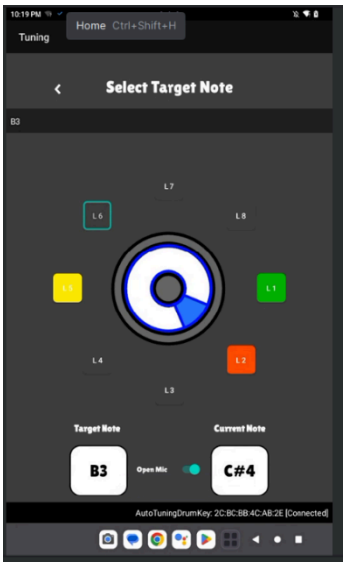


*Figure 12: This page demonstrates the color ranges of based on frequency discrepancy. The red buttons shows a lug completely out of range, yellow is in mid range, while green has hit the target frequency.*

The only features that were planned to be added was displaying the highlighted button only when lug has been tuned rather than always displaying. And displaying the target note of each lug that has already been tuned so that the user can keep track of previously tuned lugs in order that opposing lugs have the same frequencies.

# 9 Conclusion

The Auto Tuning Drum Key project successfully demonstrates the effective integration of hardware, software, and user experience considerations into a cohesive, practical tool designed for both novice and experienced drummers. By combining precise frequency detection, dynamic motor control, and an intuitive mobile interface, the device reliably automates the complex task of drum tuning, significantly simplifying and accelerating the tuning process compared to traditional manual methods. Extensive testing throughout development confirmed the system's robustness, accuracy, and usability, building confidence in its performance and potential as a valuable tool for musicians.

The finalized hardware showcases significant refinements, particularly in the compact and ergonomic enclosure that securely houses the ESP32 microcontroller, motor, microphone, and battery pack. Efforts to achieve precise internal alignment and structural integrity resulted in a visually appealing, durable product suitable for practical use. Likewise, the companion application provides clear visual feedback and structured guidance, effectively assisting users through a previously challenging and time-consuming process. The thoughtful design, reliable Bluetooth connectivity, and robust software error handling ensure a smooth and frustration-free user experience.

However, certain limitations emerged during development, especially when considering expanding the tuner's compatibility to various types of drums. Due to the distinct acoustic properties inherent in drum construction, including variations in resonant frequencies of drum shells and the tuning of the bottom drum head, the current system is specifically calibrated for the tested drum type. Extending usability to accommodate a broader range of drums is possible but would require extensive recalibration and adjustments beyond the project's original scope. Nevertheless, these limitations represent opportunities for future development rather than critical shortcomings, underscoring the importance of detailed acoustic analysis when designing generalized tuning solutions. Overall, the completed Auto Tuning Drum Key stands as a testament to meticulous engineering, thorough testing, and user-centered design, ready to support musicians in achieving optimal drum tuning results.



# 10 References

[1]“bluetooth,” *Android Developers*, 2023. https://developer.android.com/jetpack/androidx/releases/bluetooth (accessed Apr. 15, 2025).

[2]S. Jino, “The Complete Guide to Drum Tuning: Tips, Tools & Techniques,” *Equipboard.com*, Jan. 21, 2025. https://equipboard.com/posts/drum-tuning (accessed Apr. 15, 2025).

[3]*Arduino.cc*, 2024. https://docs.arduino.cc/libraries/arduinoble/

[4]“Material Design,” *Material Design*. https://m2.material.io/design/guidelines-overview

[5]H. D. Y.-D. D. Y. DIY, “Using Perfboard | Soldering Basics,” *Instructables*. https://www.instructables.com/Using-Perfboard/

[6]*Arduino.cc*, 2024. https://docs.arduino.cc/libraries/esp32servo/

[7]T. Agarwal, “I2S Protocol : Features, Working, Differences and Its Applications,” *ElProCus - Electronic Projects for Engineering Students*, Aug. 08, 2022. https://www.elprocus.com/i2s-protocol/ (accessed Apr. 15, 2025).

[8]“Envelope following,” *Ucsd.edu*, 2025. https://msp.ucsd.edu/techniques/v0.11/book-html/node153.html (accessed Apr. 15, 2025).

[9]*Arduino.cc*, 2024. https://docs.arduino.cc/libraries/arduinofft/

[10]“Guitar Tuner automatic Fender Jackson Ibanez and more,” *Guitar tuner professional. full automatic guitar tuner that fits almost any guitar like Fender, Gibson, Ibanez Yamaha*, Apr. 09, 2025. https://www.tronicaltune.net/shop/tronicaltune-plus/automatic-guitar-tuner-tronicaltune-plus-6-line-style/ (accessed Apr. 15, 2025).

[11]robrt60, “5. Drum Tuning 101 - Back to Basics ! | iDrumtune,” *iDrumTune*, Dec. 2018. https://www.idrumtune.com/5-drum-tuning-101-back-to-basics/ (accessed Apr. 15, 2025).

[12]“Harmonic Product Spectrum (HPS),” *musicweb.ucsd.edu*. http://musicweb.ucsd.edu/~trsmyth/analysis/Harmonic_Product_Spectrum.html

[13]Wikipedia Contributors, “Moving average,” *Wikipedia*, Jun. 17, 2019. https://en.wikipedia.org/wiki/Moving_average