Production Release

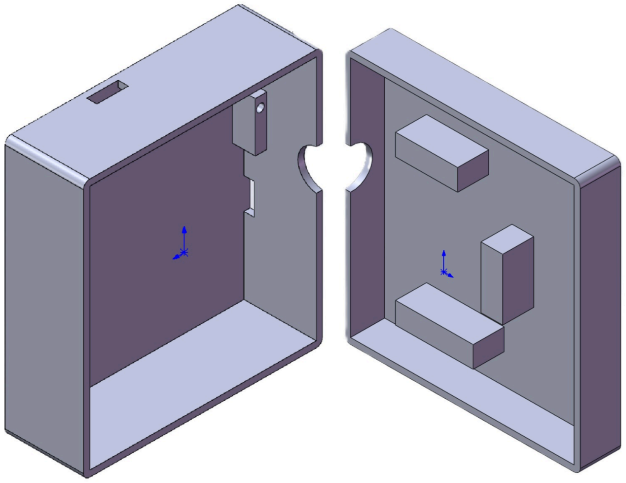
Table of Contents

Links:

Github:
https://github.com/brandonDavis264/AutoTunningDrumKey

Demo:
https://youtu.be/lw1sCPUQULc

3D Encasing:



How to Use:

1. After proceeding to the welcome page, the user will select their device and the number of lugs that their drum has.
2. Their device will connect and they will be prompted to select a target note and lug count. The device will be placed on the lug they have selected on the app.
3. Switch on open mic to start picking up frequencies as they continuously hit the drum until the target note is reached which will automatically turn off the switch in part turning off the mic and the motor.
4. The highlighted button is the indicator for what lug to tune next, making sure that the pattern being followed on the drum is relatively the same as on the app.
5. These steps will be repeated until all buttons are green.

User Experience

Interface

The production release iteration of the user interface for the application does not differ too much from the Beta version, other than a slight few additions such as a tutorial and more clear messages for the user to see the state of the device. Perhaps the biggest change is the actual device the user interacts with, the physical Auto-Tuning Drum Key itself.

As the Drum Specs page has not changed too much, we will instead talk about the Tuning Page, which has seen a few new additions. First that users will notice is a tutorial, teaching users the basic loop of the application: that is selecting a note, selecting a lug, placing the key on the lug, hitting the drum ensuring to also mute, and then repeating the process in a pattern. This is presented in the form of simple dialogue boxes going over each step. There is also an Open Mic switch, allowing the user to turn on and off the microphone whenever they want to. Lastly, some clear messages have been added if the connection between the application and the device is severed, prompting the user to return to the previous page to re-establish the connection.

As mentioned before, the biggest change made to the Interfacing of the device is the actual device itself, more specifically, the encasing of the device. This encasing has been printed in white, to make the LEDs showing the states of the device clear to the user. This includes a blue LED to confirm connection to a mobile device via Bluetooth and a green LED showing whether or not the device is powered. This allows the user to be aware of the state the Drum Key is currently in to help in case the device is debugged.

Overall, the current interface is now able to keep users more informed about the various states of the device, providing a better and more understandable user experience.

Navigation

The navigation experience has not changed much between Beta build and the current build. The main change to this is the addition of tutorials to better understand how to navigate the app, explanations of the different buttons, and what the general loop of this application should look like.
As mentioned in the previous build, the options have been pared down, and we have found that these current options are the only ones a user needs in order to get started with tuning. As such, not much has changed between the two builds within the Drum Spec page.

The Tuning page itself sees a big change within the addition of the tutorial when entering the page. Users can choose whether or not to see this tutorial with a simple yes/no prompt box. The process of selecting a lug and tuning is the same, however the addition of colored icons on the buttons help the user see if they're off (red), if they're close (yellow), or if they're in tune (green).

The user will not be able to go to another lug if the lug they're currently tuning has not been lit up green. Resetting the process is something the user can do and the indication of the next lug guides users in following the correct process for tuning the drum.

Overall, navigation through the application has been made clearer, between the tutorial explaining the loop, to the actual process of tuning the drum with the Auto Tuning Drum Key itself. While beginner percussionists may have a hard time initially figuring out what note to tune their drum in, this only needs to be taught once before the percussionist is able to navigate the application on their own.

Perception

Intuitiveness

The user interface (UI) and overall experience have been carefully designed to maximize intuitiveness through a combination of visual cues, toast notifications, and a step-by-step tutorial for first-time users. Upon launching the app, users are guided to select their Bluetooth device from a list of available connections. For added reliability, device MAC addresses are also displayed to ensure the correct AutoTuner is selected, especially in environments with multiple nearby devices.

Once connected, the user proceeds to select the drum's lug count. The interface prevents progression if any required parameter—such as lug count, device selection, or target frequency—is missing. In such cases, a toast message informs the user of the missing input. After all selections are completed, the user is presented with a top-down drum layout containing buttons representing each lug.

Selecting a lug without first specifying a target frequency prompts another toast message, encouraging proper input sequencing. Once a target note is selected, a triangular region of the drum interface highlights the area to be struck. The corresponding lug button turns red, and the next button in the star-pattern sequence is highlighted. As tuning progresses, the button color transitions through yellow toward green, visually indicating proximity to the target frequency. Upon successful tuning—defined as reaching a pitch within ±10 Hz of the target—a green indicator confirms success, accompanied by a toast notification. This cycle repeats until all lugs have been tuned, at which point no additional buttons remain highlighted.

User Goal Alignment

Once the system is paired and configured, the tuning process is both guided and automated to align with the user's goal of achieving consistent, accurate drumhead tension. Users begin by collecting the initial frequencies for each lug by striking the drum. As they designate a target pitch, the application transmits this information to the ESP32 via Bluetooth.

Users are prompted to continuously strike the indicated section of the drum while the system samples each hit. A triangular overlay on the app interface rotates to indicate which drum section to strike next, reinforcing a proper star pattern sequence. With each drum hit, the system evaluates the frequency and commands the motor to tighten or loosen the lug as needed. As the measured frequency approaches the target, the motor performs smaller corrections to avoid overshooting.

When the detected frequency falls within ±10 Hz of the target, the motor ceases operation, and the app's note indicator turns from red to green. This visual feedback confirms tuning completion for the current lug, prompting the user to move to the next. The hardware casing supports this workflow by offering wireless operation, secure motor mounting, and external access to the system's power switch and charging port.

Sensory Indicators

The system employs multiple sensory and visual feedback mechanisms to guide and inform the user throughout the tuning process. Within the mobile app:

- A triangular drum subsection is highlighted to show the active tuning region.
- The "Current Note" display updates in real time, transitioning from red to green as the detected frequency approaches the target.
- The "Target Note" remains fixed on the interface, allowing users to monitor tuning accuracy.
- A frequency value is continuously displayed at the bottom of the screen for numerical precision.
- The current button tint updates in real time, transitioning from red to green as the detected frequency approaches the target.
- The open mic switch turns off when tuning is complete.

Dropdown menus allow users to select both the number of lugs and the Bluetooth device, dynamically generating the drum interface to match the selected lug count.

In addition to visual indicators, the physical hardware provides tactile feedback. The motor's 13 kg·cm torque produces noticeable vibration during operation, which can be felt through the encasing. This natural haptic response serves as an intuitive cue—when vibration ceases, the user can infer that tuning is complete and that it's time to move to the next lug.

The 3D-printed encasing not only protects internal components but also enhances portability and usability. It includes a hardware power switch for manual control and exposes charging and programming ports for convenience and maintenance.

Responsiveness

The device delivers exceptional responsiveness, creating a fluid and efficient user experience with virtually zero lag. Every component of the app is designed to offer prompt, clear feedback, keeping users informed at every stage of the tuning process. The hardware and algorithmic design further contribute to this seamless performance—even during intensive tasks like executing FFTs to detect pitch. To enhance speed and responsiveness, the FFT was optimized to process a short segment of audio (around 23 milliseconds), enabling fast, real-time calculations. Likewise, the motor adjustments take less than a second, varying depending on the required rotation angle—typically 90 degrees or less per step.

To keep the user experience interactive and responsive, the app continuously processes frequency data in real time during tuning. Once the tuning device identifies a frequency, it sends the data to the mobile app via Bluetooth, where it’s stored in a persistent state. This allows the system to maintain the frequency information for each drum lug, ensuring consistent feedback throughout the session. During tuning, visual cues help guide the user: the frequency display changes color—red, orange, then green—as the detected frequency nears the desired target. Users initiate the tuning process manually by pressing a dedicated "open mic" button, which signals the hardware to activate the microphone and begin listening.

To avoid false starts, the peak detection algorithm only processes input when the sound level exceeds a predefined threshold. This ensures that only intentional drum strikes are captured, improving reliability and reducing user errors.

When a drum hit is detected, the device quickly analyzes the sound, determines the pitch, and sends the result to the app for immediate feedback. At the same time, the motor corrects any tuning discrepancy based on the user's selected target note. This tight feedback loop maintains accuracy and user clarity, making the tuning process smoother and more predictable. An envelope follower further filters the input, ensuring that only relevant signals—those from actual drum hits—are processed and displayed.

The latest version of the system incorporates a Proportional-Integral-Derivative (PID) control loop to enhance motor precision. This control mechanism calculates the necessary adjustments based on real-time frequency errors, allowing for quicker and more accurate tuning without relying on small incremental steps. While this increased efficiency raised concerns about potential over-tightening and drumhead damage, safety limits were put in place to address these risks. The motor is now restricted to apply corrections only within a safe range of ±90 degrees. Also the capabilities of the motor do not allow for damage to the drum’s head, instead the motor halts when faced with too much torque. Once the drum's pitch falls within an acceptable margin of error (±5 Hz), the tuning process is automatically halted, and the user is notified that the desired note has been successfully reached.

This intelligent system design ensures that each tuning step is accurate, safe, and efficient—empowering users with control while protecting their instruments and enhancing overall reliability.

Build Quality:

Robustness

Throughout the development process, our primary objective has been to ensure the robustness and reliability of the automatic drum tuning device. Both the hardware and software components have undergone extensive testing and refinement, resulting in a stable and polished final product.

In terms of hardware, our team designed, tested, and produced five iterative prototypes of the 3D-printed enclosure, each time improving tolerances, ergonomics, and functionality. The final version securely houses the ESP32 microcontroller, motor, and supporting components with enhanced structural integrity. The port orientations, internal standoffs, and the positioning of the on/off switch have been carefully optimized. Although the current enclosure aligns adequately with the hardware components, minor alignment issues remain due to the limited resolution of the 3D printer. Specifically, precise alignment between the enclosure's standoffs and the printed circuit board mounting holes could be further improved in future production runs to ensure the most secure fit and optimal protection of internal electronics.

Significant efforts have been dedicated to enhancing the robustness and reliability of the software system. The firmware running on the ESP32 microcontroller has undergone thorough refinements to ensure accurate and stable drum tuning. The microphone threshold settings have been precisely calibrated, significantly improving the accuracy of frequency detection and minimizing interference from ambient noise. Additionally, the motor control algorithm now employs dynamic rotational adjustments based on the frequency discrepancy measured during tuning. These optimizations not only reduce tuning time but also protect the drumhead from potential damage due to overtightening.

The mobile companion application has been rigorously tested and enhanced to provide a robust user experience. A critical Bluetooth connectivity issue triggered by the back button was successfully resolved, resulting in reliable and consistent Bluetooth performance. Moreover, the tuning interface now features intuitive visual feedback, clearly indicating tuning accuracy through color-coded indicators (red, yellow, and green) to assist users in assessing real-time progress. The application also highlights the specific drum lug currently being adjusted, and strict input validation requires users to define the target frequency prior to initiating the tuning process.

In the production release, the system exhibits no crashes or glitches and gracefully handles failures resulting from user errors, corrupted data, or inconsistent inputs. The robust error handling and rigorous software validation processes have ensured system stability and user trust.

While the device demonstrates significant robustness overall, it should be noted that the current microphone threshold and motor adjustment algorithms were primarily tuned for a specific drum type, and may require minor calibration for optimal performance on drums with different acoustic properties. Overall, extensive iterative development and meticulous testing have resulted in a robust and reliable drum tuning device that delivers accurate tuning, enhanced user experience, and dependable operation.

Consistency

This project is built around a singular, cohesive goal: providing a smooth and consistent user experience across all platforms. At the heart of the design is a commitment to simplicity—every component is crafted to improve user interaction and streamline the drum-tuning workflow.

The mobile app acts as the central hub for engaging with the tuning hardware. Its interface is intentionally minimal and user-centric. Users only need to input the essential details, like the number of drum lugs and the desired pitch. A visual representation of the drum, complete with highlighted segments, helps users easily identify the currently selected section and the active lug being adjusted. This approach keeps input requirements simple while retaining the accuracy needed for effective tuning. The software and hardware take care of the complex processing behind the scenes, enabling users to tune their drums quickly and effortlessly.

Connecting the app to the tuning device is designed to be intuitive and dependable. Users power on the device with a built-in switch, then connect through their phone's Bluetooth settings. Once paired, the app automatically transitions to the tuning interface, ensuring users are only able to proceed once a stable connection is in place. This helps eliminate confusion, with the main screen clearly instructing users to establish the Bluetooth link before moving forward.

The tuning screen provides real-time feedback, displaying live frequency data and notifying users when the system is ready to capture a drum hit or detect pitch peaks. This ensures an accurate, responsive tuning session with minimal guesswork.

In the latest iteration of the design, once a drum hit is detected and its frequency is measured, the system automatically adjusts the tension of each lug using a precise PID control loop. This loop calculates and applies the required correction based on the detected frequency. As each lug reaches the target tuning, the user is alerted—either to move the device to the next lug or that tuning for the current lug is complete. The app continuously tracks the tuning status, displaying saved values for each lug, eliminating uncertainty about what has been tuned. This structured process ensures consistent and professional tuning results, reinforcing the system's reliability and ease of use.

By prioritizing consistent design and smooth operation, the entire user journey—from initial setup to final tuning—is made efficient, intuitive, and cohesive.

Aesthetic Rigor

Throughout the development process, significant emphasis was placed on ensuring the automatic drum tuning device not only delivered optimal functionality but also presented a refined and professional appearance. The finalized enclosure design embodies a carefully optimized form factor that balances compactness, ergonomics, and visual appeal. Multiple iterations of the enclosure allowed our team to incrementally enhance tolerances, improve structural integrity, and streamline the positioning of internal components such as the ESP32 microcontroller, motor, and USB-C port.

The production enclosure successfully eliminates unnecessary bulk while maintaining durability and ease of handling. Particular attention was given to ensuring precise alignment and seamless integration of the USB-C port, internal PCB standoffs, and the on/off switch. These adjustments have significantly improved usability and provided a clean, professional aesthetic. Minor alignment imperfections related to the precision of 3D-printed standoff holes have been minimized as much as possible given current printing capabilities, and any previously noted difficulties in cleanly removing supports for the on/off switch have been addressed through refined printing techniques.

In parallel, the mobile application's interface has been rigorously refined to ensure clarity, intuitive navigation, and aesthetic consistency. The tuning screen incorporates clear and immediate visual feedback through a color-coded system (red, yellow, and green) that effectively communicates the tuning accuracy relative to the target frequency. Additionally, the user interface clearly highlights the specific lug currently being adjusted and consistently guides users through a star-shaped tuning sequence to ensure uniform drum tension. The integration of these visual aids results in an organized, straightforward user experience, enhancing both usability and aesthetic appeal.

The final production release exhibits no cosmetic defects or visual inconsistencies. All graphical and auditory assets have been carefully curated and tested, ensuring they are fully functional and seamlessly integrated within the application. The consistent visual design, clarity of feedback mechanisms, and polished interface collectively ensure that aesthetic considerations complement and enhance the device’s overall functionality, providing users with a satisfying and professional experience.

Vertical Features

External Interface

User Interface

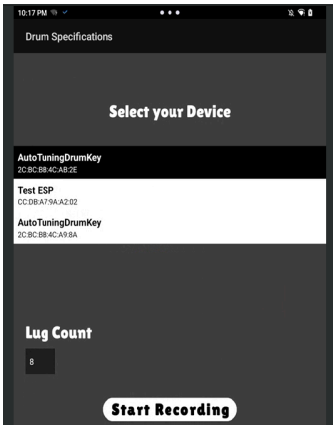
The core user interaction occurs through the companion mobile application, which provides a streamlined and intuitive interface. Hardware interaction is minimal and limited to powering on the device and physically clamping or holding it in place during tuning. Within the app, users specify the drum's lug count, select the desired target note, and begin tuning by striking the drum. The system then automatically adjusts each lug in real time as the user continues to strike the drum surface.

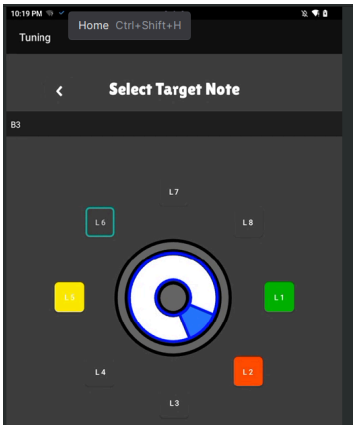
The application interface features a triangular guide that visually directs users to strike specific drum sections in a star pattern. Highlighted buttons appear in the app to indicate which area to hit next. Upon reaching the target frequency, the corresponding button transitions from red to green, and the blue LED on the ESP32 turns off, signaling that tuning for that lug is complete. Users are then prompted to reposition the tuner to the next lug, continuing the process until all lugs on both the top and bottom heads are tuned.

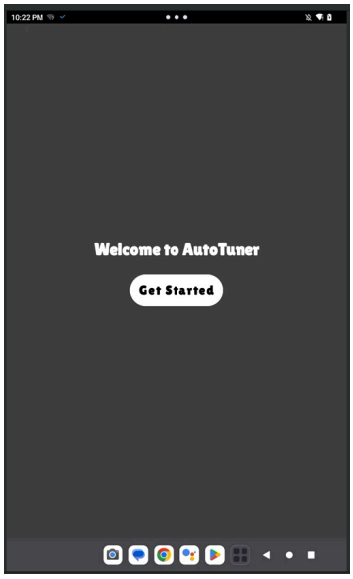
App Interface and Controls

The application is designed with a focus on flexibility and user guidance. Users can dynamically select the number of lugs, choose a connected tuning device via a dropdown list, and view real-time feedback as the system detects and displays the current note. Each lug's status is visually indicated: when a lug is out of tune, it appears red, and once the target note is reached, it turns green. The current frequency is also displayed in a text field, color-coded from red to green based on proximity to the desired pitch—green representing near-perfect alignment.

The app supports a structured tuning process by guiding users through a star-shaped tuning sequence, helping distribute tension evenly across the drumhead. For first-time users, a built-in tutorial explains the device setup, usage flow, and tuning methodology, ensuring accessibility for both beginners and experienced musicians.



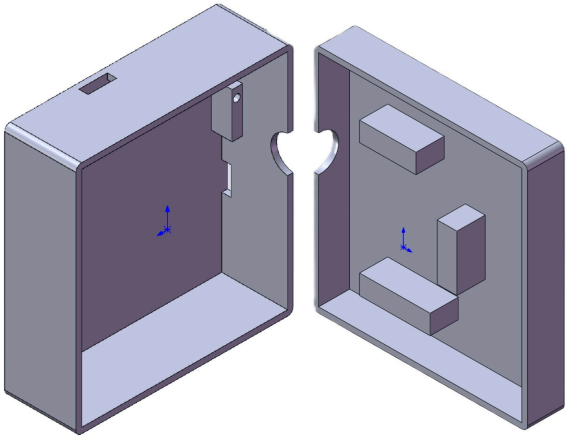


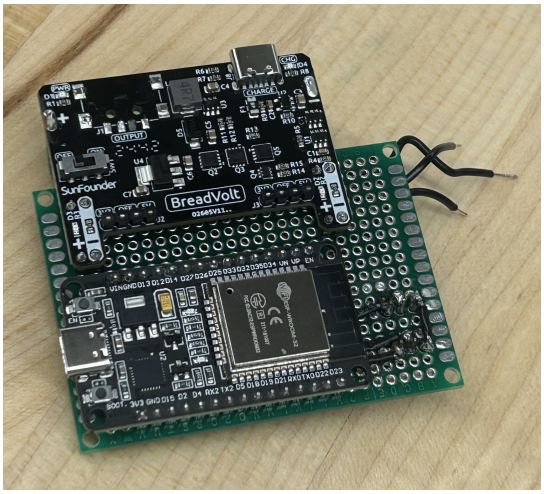


Hardware Interface

The hardware integrates the ESP32 microcontroller, I2S microphone, and TD-8135MG servo motor on a compact perfboard layout. Key interface components—such as the power switch and charging port—are externally accessible to facilitate usability. The encasing protects the internal electronics and contributes to the device's portability and mechanical robustness.

The motor provides 13 kg·cm of torque, sufficient for rotating most standard drum lugs. A plastic drill bit is adhesively affixed to a motor shaft adapter, which is itself securely bonded to the motor, allowing effective and controlled rotational adjustments during tuning.





Persistent State

Software Persistent State

Although the current implementation does not include a backend or cloud-based database, future versions of the application plan to use Android's SharedPreferences library to persist user settings—such as lug count and preferred notes—across sessions. These settings will remain local to the device and restore automatically upon reentry.

Hardware Persistent State

The ESP32 processes real-time audio data from the I2S microphone, applying FFT and Harmonic Product Spectrum (HPS) algorithms to detect fundamental frequencies. The motor logic is governed by frequency error mapping, translating the difference between detected and target frequencies into angular motor adjustments. A moving average filter helps suppress noise, and motor correction ceases once the detected frequency falls within an acceptable tolerance

band. No haptic feedback is currently implemented to signal tuning completion, but visual indicators within the app serve as effective cues.

Persistent State

For the persistent state of the automatic drum tuner, here is lots of data that is transferring over a classic bluetooth connection. Including the target frequency the auto tuner will tune to and the frequency the I2S microphone picks up. Currently most of the persistent data that is being used to tune the actual device is on the ESP32 rather than the companion app.

Software Persistent State

The Android companion app provides a user-friendly interface that interacts with the ESP32-based tuner hardware. It does not process frequency data but instead acts as a conduit for sending commands and receiving feedback. The primary function of the app is to allow the user to select a target tuning note (currently limited to 25 different notes) through a dropdown menu, which sends the frequency to the ESP32 via Classic Bluetooth. The dropdown is designed to support up to 25 notes, and the communication infrastructure has already been tested and works.

Additionally, the app displays the detected frequency from the ESP32 in real-time and uses color indicators to show whether a lug has reached its desired note. A notable issue in the beta version is a crash when the lug count is set to zero. This is caused by the UI attempting to generate a lug diagram without any lugs, which was resolved by implementing a simple try-catch block to prevent application termination.

When a valid lug count is entered, the app dynamically generates buttons representing each lug. This allows the user to visualize the tuning status of each lug and ensures a consistent UX.

There is a mic switch to turn off and on the microphone when the user finishes the tuning of a lug. They would take it off and put it on another. Then, the user will click the “mic on” switch to turn on the mic again via bluetooth

Hardware Persistent State

The ESP32 is responsible for handling all the core signal processing and motor control logic. It reads microphone data using an I2S microphone interface and applies a layered filtering and analysis pipeline. This pipeline includes:

- Envelope Follower: Detects when a drum hit exceeds a set threshold (magnitude > 300).

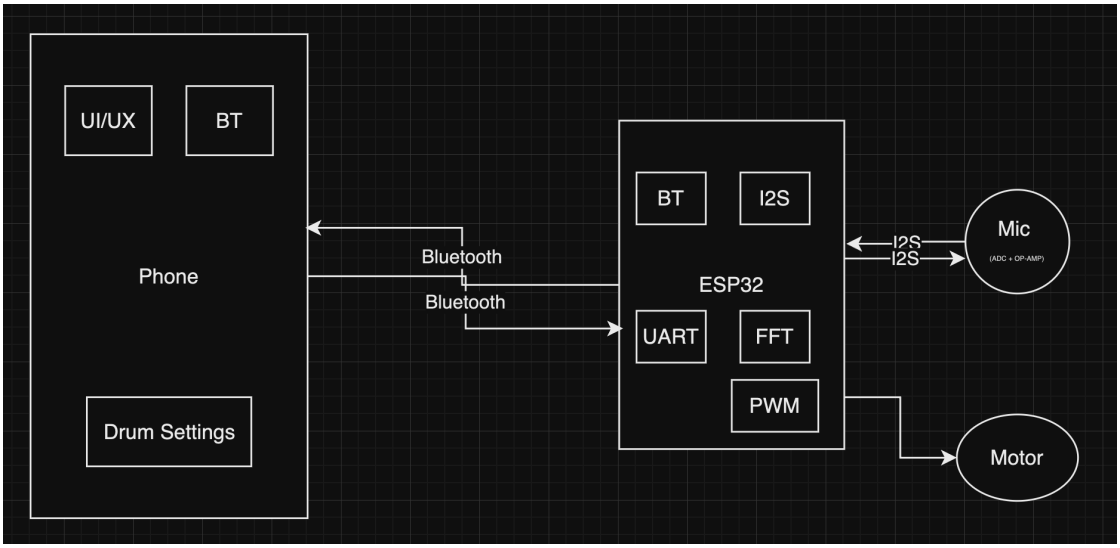
- FFT (Fast Fourier Transform): Converts time-domain mic data into frequency domain.
- Moving Average Filter: Smooths out raw frequency values over multiple samples.
- Harmonic Product Spectrum (HPS): Refines pitch detection, especially useful for drums with complex overtones.

Once a qualifying impulse is detected, the ESP32 calculates an average frequency and sends it over Bluetooth to the companion app. The device then compares the detected frequency against the target frequency (250 Hz ±10 Hz). If the value falls outside this range, the servo motor rotates 180 degrees clockwise to tighten or counterclockwise to loosen the lug. If the frequency is within the ±10 Hz tolerance, the motor remains stationary.

To prevent memory leaks, FFT audio samples are stored in heap memory and cleared after use. The system uses stack or RAM memory to store the buffer size and state variables.

Full Data Communication

The data transfer model bellow is how all the data is currently being transferred in the system and communicating with all the devices



For Bluetooth:

- Data for the target frequency is being written by the phone to the ESP32
- Data from the ESP32's microphone is being read by the phone

For the I2S:

- An I2S datastream is being read by the ESP32 and converted to frequencies by an FFT if the threshold is over the microphone's filtered data through an envelope follower, harmonic product spectrum, and moving average filter.

For PWM:

- The target frequency will determine the time to write to the continuous motor through pulse with modulation and timers on the ESP32

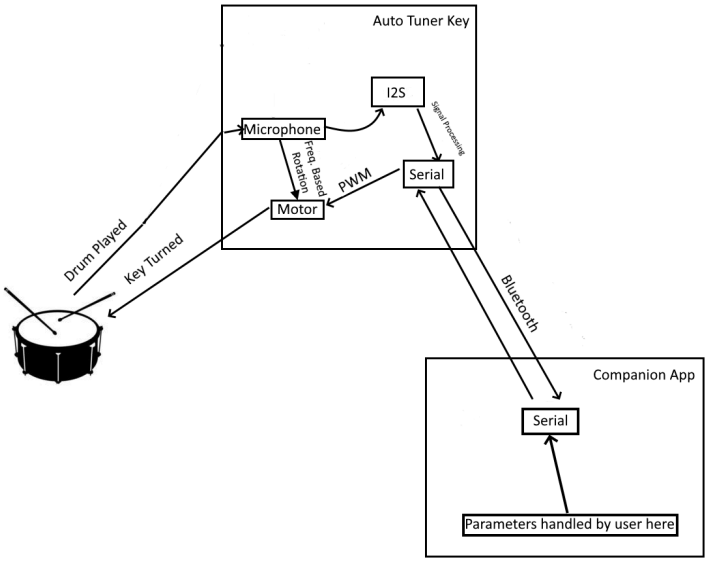
Internal Systems

Component Architecture

The system consists of:

- **Hardware:** ESP32 microcontroller, I2S microphone, LED indicator, FFT processing, servo motor, and a battery pack, Moving average filter, Harmonic Product Spectrum (HPS) algorithm.
- **Software:** Android app for user interaction and Bluetooth communication. User Experience and Interaction, a microphone switch to turn on the microphone

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.



System Overview

The system consists of 2 major components:

- Hardware: ESP32 microcontroller with an I2S microphone, LED indicator, FFT processing capabilities, Harmonic Product Spectrum (HPS) algorithm, Moving average filter, envelope follower for peak detection, a moving average filter to attain a frequency of an impulse, a continuously rotating servo motor, and a battery back to power the ESP32 (3.3V) and the motor (5 V) separately.

- Software: Android mobile application is mostly for the user interface experience as well as to control the servo's target frequency to stop tightening or loosening. The companion app will also get the impulse response frequency from the microphone and interpret the data to show in an understanding way to the user. There is a microphone on switch with the UI to turn on the mic via bluetooth on the ESP32.

The ESP32 continuously listens for sound impulses, filtering them through an envelope follower before applying FFT and HPS algorithms. The firmware uses BluetoothSerial to receive target frequencies and send back detected values.

The app initiates the tuning process by turning on a switch to begin recording. As frequencies are detected, they are displayed visually, and the app tracks tuning status lug-by-lug. Servo motor control is currently hardcoded to rotate 180°, but future updates will scale the rotation based on proximity to the target frequency.

Hardware Components

The ESP32, microphone, and motor are compactly mounted on a perf board. External switches and charging ports are repositioned for accessibility. The 3D-printed enclosure not only protects the internals but also makes the device portable and user-friendly. The servo motor provides 13 kg*cm torque, more than sufficient to adjust drum lugs reliably.

ESP32 Microcontroller:

- Configured to process I2S microphone input.
- Performs real-time FFT analysis using the ArduinoFFT library.
- Moving average filter
- Harmonic product spectrum
- Implements an envelope Follower for peak detection
- Communicates with the Android app via Bluetooth using BluetoothSerial.
- LED (Pin 2) indicates Bluetooth connection status.

I2S Microphone:

- Pins:
  - WS (Word Select): GPIO 4
  - SD (Serial Data): GPIO 15
  - SCK (Serial Clock): GPIO 23
- Connected to the ESP32 I2S interface for audio data acquisition.

Servo Motor:

- Pins:
  - Data Line: GPIO 18
- Connected the ESP32 for continuous rotation inorder to twist a lug at tensions less than 32 kg*cm

Productization:

- PCB design for the component
- Solidworks design for 3D enclosure
- 3D printed drill bit to attach to the servo motor to twist a lug

Power Supply:

- A bread volt battery backpack for arduino that powers the ESP32 at 3.3V and the continuous servo motor at 5V.

Software Components

ESP32 Firmware

- Configures I2S to read audio samples.
- Processes audio data using FFT to extract frequency peaks via envelope follower.
- Even out the signal from outliers with a moving average filter and Harmonic product spectrum
- Communicates with the Android app over Bluetooth to send calculated frequency values.
- Listens for mic on switch to start recording from the Android app.
- Rotates a continuous servo motor +/-180 to tighten or loosen a lug

Android Application

- Uses Bluetooth APIs to establish and manage communication with the ESP32.
- Displays a user-friendly UI with spinners for drumhead and shell configurations.
- Has a mic on switch to turn on the mic on the ESP32 via bluetooth
- Sends a target frequency to tune a drum to.
- Displays the calculated average frequency received from the ESP32.

Permissions:

- Bluetooth connect, scan, and location permissions are included to ensure compatibility with modern Android versions.

Major Libraries and Frameworks

Arduino Framework:

- Used for ESP32 firmware development.
- Includes BluetoothSerial, driver/i2s.h, and arduinoFFT.h., servo.h

Android Framework:

- Provides UI components and Bluetooth communication APIs.
- Uses ActivityResultContracts for managing runtime permissions.