Automatic Guitar Tuner

Adam Harmon, Bryan Casey, Jason Lupo, John Geiger

Department of Electrical Engineering and Computer Science, University of Central Florida, Orlando, FL

***Abstract* - This document describes the design of the handheld, and rechargeable, Automatic Guitar tuner, which includes the subsystem designs and specifications. The aim of the project is to create a device to allow beginner guitarists the ability to easily and quickly tune their guitar. The project involves the utilization of signal processing, signal amplification, Bluetooth integration, integrated circuits, and an Android user interface. All of these processes are used together to tune an individual guitar string to a user selected note, which is done on the Android mobile application. The result of this project is an accurate handheld guitar tuner, along with an easy to use mobile application for Android phones.**

I. INTRODUCTION

Tuning a guitar has been a difficult process for beginner guitarists since the creation of the instrument. Only recently, the technology required to implement a more nuanced solution to this problem has become common. The continuity of Moore's Law and increasing access to powerful, yet cheap integrated circuits and microprocessors has made the dream of an automatic guitar tuner a possibility instead of just a dream. An automatic guitar tuner allows new musicians to tune their guitars with computer numeric control (CNC) precision, while an application allows them to learn what each note should sound like. The device also has applications for professionals like teachers who would like a faster way to tune multiple guitars for their students.

This project allows the user to use all the main operating modes without an external application while also providing more functionality through an application if desired. The system will have a display to provide menus and basic feedback, while providing multiple buttons with intuitive control directions given on screen.

One of the most difficult aspects of this project is the integration of frequency sensing and physical tuning. There are many ways to extract the core frequency of a musical instrument which range in cost of application. The control system behind the tuning servo also needs to be as fast as possible, while still retaining accuracy to ensure ease of use. A microcontroller will be used in conjunction with sensors, ADC converters, motor drivers, and the OLED display to provide end results to the user. The device will also feature a rechargeable battery system which will cut costs in batteries when the device is used heavily. The device will charge off a simple 5V power supply and handle all battery management internally.

The resulting system that has been designed is accurate, and easy to use. It will provide features to be used by beginners and professionals alike. Along with its utility in quickly tuning guitars, the device will also be useful as a learning aid to people who want to improve their understanding of each note and how to manually hear the small differences in the sound.

II. SYSTEM COMPONENTS

The design is best presented in terms of system components, which are the physical individual modules that are interfaced to create the final project. This section provides a overview of each component.

A. Bluetooth

The Adafruit Bluefruit LE UART Friend Bluetooth module was chosen to use for the wireless communication between the microcontroller and an android cell phone. The selected Bluetooth module was an ideal choice as it's low cost and uses low energy.

B. Microcontroller

The brains of this project is the Teensy 3.2 development board. The Teensy board was chosen for its high clock rate, large memory, and its community support, as it can use the Arduino libraries.

C. Microphone

The selected microphone, the PMO-6027P-40KDQ, is a microphone manufactured by Mallory Sonalert Products Inc. This microphone is a condenser microphone that fulfills the pass band requirement necessary to pick up the most common string instruments. This microphone is an ideal choice as it's low cost and has good signal to noise ratio, diameter, and depth.

D. Motor

Motor selection for physical tuning of the peg needed to be done carefully to provide adequate speed as well as torque. The motor that ultimately satisfied the requirements for this project was

brushed DC motor with 1:100 planetary gearbox capable of rotating at 60 RPM at 6V input.

E. Operation Amplifier

Before the microcontroller on the automatic guitar tuner receives an analog signal, it needs to be amplified. The design for this project is going to need a couple of Op-Amps in order to design filters, amplifiers, and other electrical equipment. The TL084CN Op-Amp, manufactured by Texas Instruments, was selected as it's able to run off a single power supply, and it has enough channels needed for this project, which will save space on the printed circuit board(PCB).

F. Power

The Automatic Guitar tuner is powered by two lithium ion batteries. To be rechargeable a battery charging IC is needed to charge the batteries. The BQ2057CSN chip was selected for this task, as it is capable of adequate charge current, cell conditioning, and temperature monitoring.

G. Android Application

The user interface for this project is an Android Application.

The app uses the mobile phone's built-in Bluetooth capabilities to connect to the Bluefruit LE UART Friend on the automatic guitar tuner. The application allows the user to select what note they want a guitar string to be tuned to. The application can also save tunings, giving the user a faster way to tune each guitar string if they're going to tune a string to a note they have tuned to previously.

Fig. 1: Android Application Menu

H. OLED Display

The OLED display provides the user with with concise and informative data regardless of whether the device is operating in manual or automatic modes. The display also provides the current state the device is in (Idle, Ready, Tuning, etc.). The Adafruit Monochrome 0.96" 128x64 OLED module was chosen for this task.

III. SYSTEM CONCEPT

To better understand how the system works as a whole, a flowchart of how the automatic tuner works is shown below.

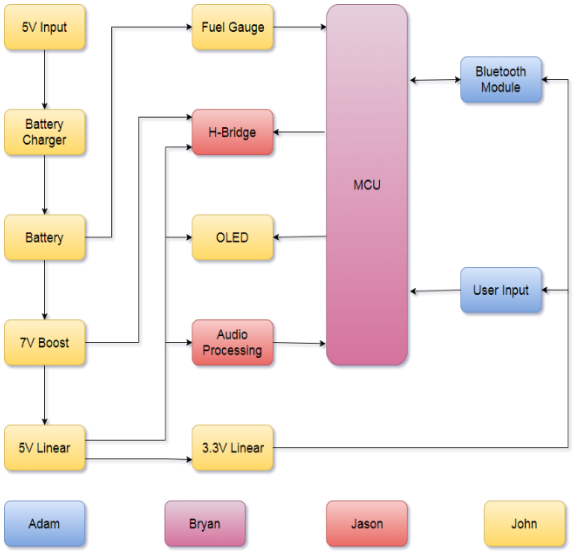

Fig. 2: Complete Flowchart of the system showing how the systems are linked together.

As seen in the flowchart above there are seven main components to the system. The user will be utilizing only two parts of it, creating the Bluetooth connection between the android phone and the tuner, and the button input.

The mobile application will allow the user to select the desired note that is wanted. Once this is done, the user will pluck the desired guitar string and the microphone will receive the sound. This information is then sent to the MCU, which will gather the frequency of the plucked string and communicate with the motor. which will either receive a clockwise or counterclockwise signal from the MCU's frequency algorithm to tune the guitar string.

A. Microcontroller Concept

The microcontroller is the core of the tuner system. The MCU is responsible for outputting and receiving data from the various subsystems. When applicable,

received data is recognized as meant for control via the Button Input subsystem or Bluetooth Communication subsystem, output to another subsystem such as the Bluetooth Communication or Backlit Display subsystems, or input data from the Rechargeable Battery and Power Regulation, Bluetooth Communication, or Sound Feedback subsystems.

The MCU contains the code used for tuning the guitar. In the tuning code there's a frequency detection part used to detect the frequency of the plucked guitar string, and comparison part to compare this fread in frequency with the desired frequency the user wants. The MCU also contains the code for Bluetooth receive and transmit data.

B. Power Concept

After charging, the battery is tasked with providing power to the entire system, including the motor and all of the logic. Originally, this project intended to use a single lithium ion cell to power the full system. This was changed to use two lithium ion cells in series to provide a 7.2V nominal voltage. The following linear regulators were also selected for their ease of implementation in the system and low common mode noise.

Function	Manufacturer	Part Number
Linear Regulator 5V	STMicroelectronics	LM7805CV
Linear Regulator 3.3V	STMicroelectronics	LD33V

Table 1: Power Regulation ICs

C. Tuning Concept

To tune a guitar string, the user first needs to select a note to tune to on the Android Application. Once this is done, the desired string can be plucked, and the MCU will determine whether the string needs to be tuned or not. If it does the MCU will tell the motor to either turn clockwise or counterclockwise. When the motor has turned the appropriate amount, the MCU will tell the motor to stop. The string can then be plucked again, if it still needs to be tuned the process will be repeated, if not then the motor won't turn and the string is in tune to the selected note.

IV. HARDWARE DETAILS

A. Bluetooth Module

Along with the built-in capabilities of Bluetooth on an Android cell phone, the Adafruit Bluefruit LE UART Friend is used by the Teensy 3.2 to allow serial communication from the Android Application. The module operates at 3.3V, and consists of VIN, GND, TX, and RX pins. The TX pin transmits data to the Android application, and the RX pin receives data from the Android application. The module blinks red when it's looking to pair with a device, and it blinks blue once a connection is formed between the Bluetooth module and a device, in this case a cell phone.

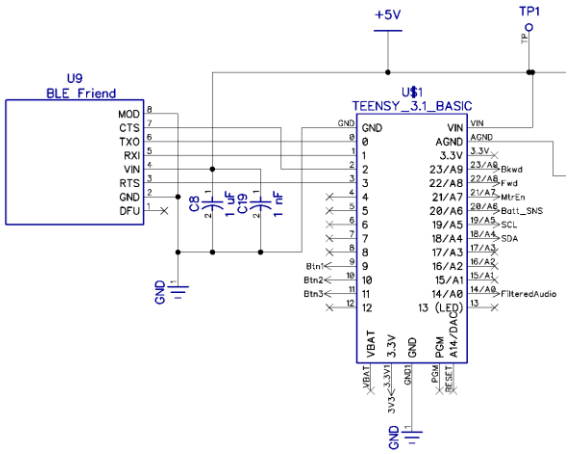

Fig. 3: Bluetooth Module Schematic

B. Microcontroller

Originally, the MSP430G2553 was chosen as the primary MCU for this project. After doing proper research on frequency extraction techniques, it was noted that an MCU with more RAM was needed. The Teensy 3.2 32-bit, 72 MHz microcontroller with 64 kBytes RAM. It is a complete USB-based microcontroller development system which allows us to quickly make changes to code and upload it to the device. The onboard ADCs are capable of 12-bit conversion which eliminated the need of the previous external ADC from the V1 design. The development board also contains an onboard 3.3V regulator which is capable of powering the button and I2C inputs. The Teensy 3.2 is perfect for the rapid prototyping and improvement of this device.

C. Motor

Motor selection for physical turning of the peg needs to be done carefully in order to provide adequate speed as well as torque. One of the desired features of this device is to allow for easy de-stringing and restringing of the guitar so a minimum rotational speed of 30 RPM was chosen to keep this functionality practical. The minimum torque was determined by physical testing on a guitar.

Torque is a measure of rotational force, measured in units of force*length. In order to measure the needed torque to tune a knob on the guitar, the peg was attached to a lever arm and a calibrated scale was used to measure the force needed to rotate the lever. The following equation was used to calculate the torque being applied.

 (Torque) = F * D (Force * Distance)

String	Distance (in)	Force (oz)	Torque (oz-in)
1	6	2.1	12.6
2	6	2.5	15
3	6	3.8	22.8
4	6	2.7	16.2
5	6	3.1	18.6
6	6	4.5	27

Table 2: Torque Requirements

Since testing shows that a value of 27 oz-in is required to rotate the thickest string on a 6 string guitar, a minimum output torque of 35 oz-in was chosen to guarantee adequate tuning across all guitars. The motor that ultimately satisfied the requirements for this project was a generic 1:100 brushed planetary gearbox motor.

D. Audio Signal Processing

The microphone selection needs to be carefully considered to ensure that the device can accurately determine the frequency at which the guitar string is tuned to. This is base in which the rest of the design will be built off of. It needed to have a pass band that would be able to detect all possible frequencies a guitar could be tuned to as well as having the ability to continue development and tune other instruments as well.

The PMO-6027P-40KDQ microphone is a condenser microphone with a pass band from 20Hz to 16KHz. The chosen microphone also has a sensitivity rating of -40dB and a signal to noise ratio of 56dB. This will ensure the signal is strong and clear.

The signal produced by the microphone will then be amplified and filtered before being sent to the microcontroller's analog to digital converter.

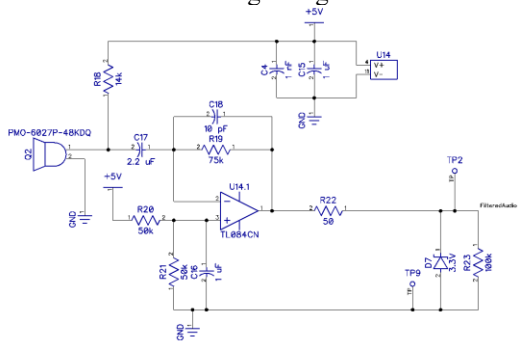

Fig. Audio Signal Processing Schematic

E. Power

After testing and troubleshooting, it was deemed necessary to move to a 2s1p configuration to eliminate noise from a boost regulator in the previous circuit. The nominal 7.2V from the 2s configuration is adequate to power the motor and two linear regulators listed below. After hardware changes from V1 to V2, the 3.3V regulator is not strictly necessary, as the Teensy 3.2 is capable of creating its own 3.3V line, thus the LD33V socked is left unpopulated. The 5V regulator powers all logic components of this project.

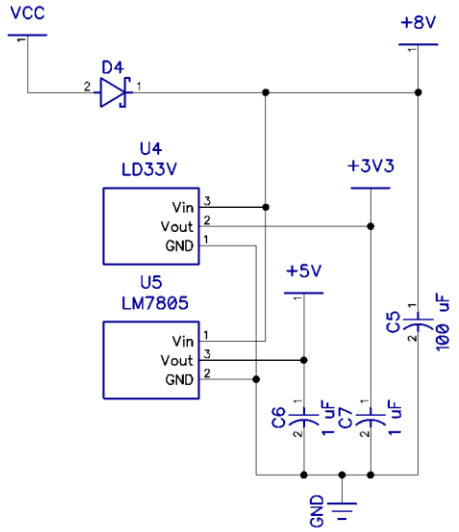

Fig. 4: Power Regulation Schematic

F. OLED Display

The OLED display shows the user information data about the tuning. It provides the currently

received frequency is displayed, along with the target frequency. The time taken to complete a full string tuning is displayed once automatic tuning completes. If operating in manual mode, the current direction, clockwise or counterclockwise, the motor is turning is shown.

The figure below shows how the OLED display connects to the MCU via I2C. The display is powered from the 5V bus, same as the MCU. The two components share a common ground which is needed to allow for low noise communication over the I2C bus. The display is tested by running an example Arduino script to send text and figure to the display.

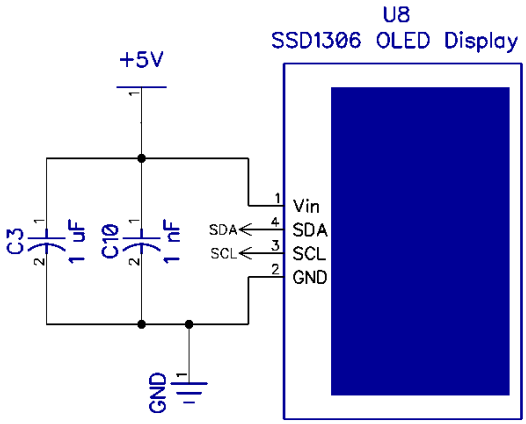

Fig. 5: OLED Display Schematic

V. SOFTWARE DETAILS

A. Android Application

The Tuner Application is available to download for free via the Google Play Store for all Android devices running Lollipop OS or any more recent Android OS release. The app will communicate via Bluetooth tether with the physical tuning device. Intended use of the app is as both a wireless control and additional information display for the tuning device. The app will consist of five screens, the Main Menu, Tuning, Tuning Library, and Options screens.

The Main Menu screen displays the currently selected tuning and the status of the Bluetooth connection between the application and the device. This screen is displayed by default when the application is first opened and may be returned to via a button on each of the four other screens. In addition, the main menu provides navigation points to the other four application screens. Finally, an exit button immediately shuts down the application and break the Bluetooth tether.

The Tuning screen is where all active communication between the application and the physical tuning device takes place. The active tuning is displayed to the user, this includes all strings involved in the tuning along with the specific frequency each string is to be tuned to. The user may then select an individual string, and lock it in for tuning. After locked in, the string's tuning frequency is sent from the application to the physical tuner.

Once the frequency is received by the tuner, the device waits for the user to activate tuning via physical button input. On activation, the device sends a signal to the application which then prompts the user to strum the string. The user may be prompted to strum multiple times until the intended tuning frequency is achieved. On tuning completion, statistics from the tuning session are displayed to the user including starting frequency and the time it took to tune the string. The user may now choose a new string to tune and the tuning process begins again.

The Tuning Library screen consists of a local database of standard and user defined tunings. From this screen, the user may select a pre-existing tuning to become the active tuning, edit a pre-existing tuning, delete a preexisting tuning, and/or create new custom tunings.

Creation of a custom tuning begins by selecting the "New Custom Tuning" button. On selection, a UI form is displayed consisting of a user input box for the name of the new tuning, a button to add a new string to the tuning, and a "Save" button. When a new string is added, the existing form is expanded to contain a user input box for the string's name and that string's frequency. Once the user has completed the form in total, the "Save" button may be pressed to save the custom tuning to the local database and the user is returned to the Tuning Library screen.

Editing or activating a pre-existing tuning begins with selecting a tuning by tapping on the title of a displayed tuning. Once selected, the tuning's string titles and frequencies are displayed, along with an "Edit" button, a "Delete" button, and a "Set Active" button. The "Set Active" button will set that specific tuning as the active tuning which is used during the tuning process. The "Delete" button prompts the user with a dialogue asking if the user is sure he/she wishes to delete the selected tuning. The user may choose to accept the deletion, which will permanently delete the tuning from the database, or to cancel the deletion in which case the tuning is preserved. The "Edit" button works similarly to the creation of a custom tuning, the same UI form is displayed as in the custom tuning process with the difference that some input boxes will already be filled out with information that the tuning already possessed.

B. Code

The code for the Android application is developed primarily in the Java programming language. This is a design constraint enforced by all Android operating systems. The application is developed using Android's official IDE, Android Studio. The Extensible Markup Language (XML) is also employed.

The coded flow chart seen in the figure below shows the brainstormed plan for menu navigation on the physical device. This menu will be viewed through the OLED display and navigated using buttons on the device. The menu will flow with a vertical scroll interface, using buttons for up and down movements, respectively. There is a center select button which will move further down the branch of menu selections. The flow chart starts with the initial opening of the interface by turning the device on. The main menu contains options for choosing Tuning Type, Adding Custom Tuning Type, Manual Tuning Mode, and Exiting. Each of these menus has further selections which can be made and offer a great range of functionality to the user.

The Choose Tuning Type branch allows the user to select the particular note which is desired for the string in question. This menu option can also store multiple tuning profiles, where all the strings can be tuned in sequence without needing to select the notes individually. Pressing the select button on the chosen Tuning Type will prompt the user to place the device on the guitar knob and strum the string. The device will automatically extract the carrier frequency in the sound of the string and tune the string to the desired note.

The Add Custom Tuning branch allows more flexibility in this device over the default profiles included in the memory. The user selects this menu and can create a new profile. The user selects the number of strings to be stored in the profile, then inputs the desired note for each string. The note can be input as a classical letter, or if the corresponding frequency is known, the numerical frequency. After the parameters are set to the user's preferences, the profile is saved and stored on the device memory for later use.

The Manual Tuning Mode branch can be used for a few functions which give greater control to the user. The user can input a desired note to be tuned to and press the select button. The device will then listen to the sound output of the string and display the corresponding note/frequency of the string. The user can then manually turn the knob and observe the frequency as it changes. In a way, this mode allows the user to train their 'ear' and learn the difference in notes and how to manually tune the guitar on their own. This menu also offers a Hold to String or De-string option which will spin continuously while the user holds the up and down buttons respectively. This function saves the user from having to repetitively turn the knob when changing strings on the instrument and prevents over tightening when restringing.

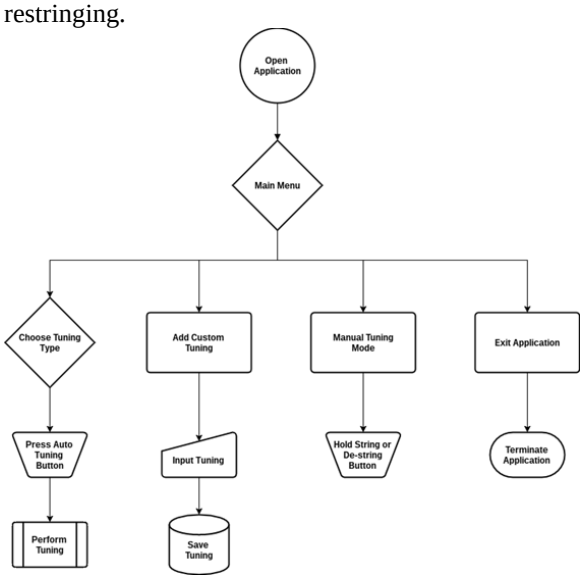

Fig. 6: Coding Flowchart

VI. BOARD DESIGN

The PCB for this project was designed entirely using Diptrace. This program is an electrical CAD program which allows engineers to design everything from individual components, to schematics, to full PCB layouts. The entire design was brainstormed and then the schematic created in Diptrace.

All schematics were breadboarded and tested separately, then the full schematic exported to the layout editor. The physical board was designed to separate the high power components from low powered components as much as possible. The charging port, battery input, and motor controller were all placed near the top, the microcontroller placed near the middle, and the audio preamplifier placed at the bottom. The large separation between power and audio should provide for the cleanest signal for frequency extraction.

The digital PCB layout was exported to gerber and drill files and a physical copy was ordered from OSHPark. The boards are manually populated, using mainly surface mount passive components and through hole active components. This combination allows an easily tested board while still retaining a small physical size necessary for a handheld device.

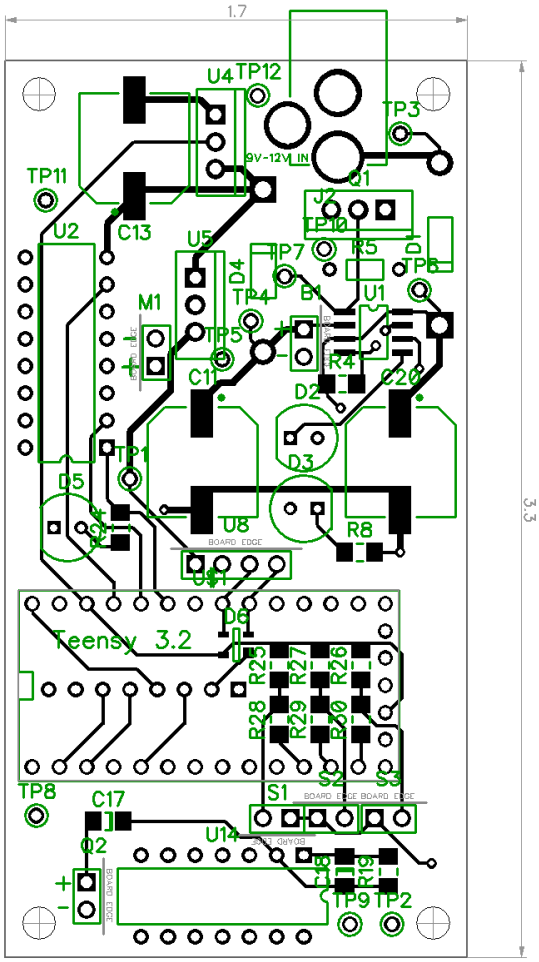

Fig. 7 PCB Layout

VII. CONCLUSION

The Automatic Guitar Tuner was a complex but enjoyable project. It allowed us to implement all the knowledge we have learned through our time here in the College of Engineering at UCF into one final project.

All of these subsystems are integrated in such a way to ensure the best possible automatic guitar tuner was constructed. This included redesigns of subsystems as efficiencies were found. This ensures a final and completed projected.

While most of the group are not musicians, we have all come to appreciate the effort and complexity of audio systems. This inexperience with musical elements allowed us to approach this project through the eyes of a beginner and ensure simple to learn and help develop the user as musicians.

THE ENGINEERS

Adam Harmon is a 23-year graduating with his Bachelor's degree in Electrical Engineering. He is looking for positions within the renewable power industry, specifically solar power.

Bryan Casey is a 24-year old graduating with his Bachelor's degree in Computer Engineering. He is looking for positions within the artificial intelligence field or the CPU architectural design field.

Jason Lupo is a 22-year old Graduating Electrical Engineering Student. He is looking to get into the automation industry, including integration of PLC systems. This interest stems from his internships with Walt Disney World Parks and Resorts.

John Geiger is a 24-year old graduating with his Bachelor's degree in Electrical Engineering. He has accepted a position with Texas Instruments as an Applications engineer, and is looking forward to moving to Dallas, TX to begin his career.

ACKNOWLEDGEMENT

The authors wish to acknowledge the assistance and support of Dr. Lei Wei, Dr. Samuel Richie and the University of Central Florida.