5 Testing

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, power system, or software.

The testing plan should connect the requirements and the design to the adopted test strategy and instruments. In this overarching introduction, given an overview of the testing strategy and your team's overall testing philosophy. Emphasize any unique challenges to testing for your system/design.

In the sections below, describe specific methods for testing. You may include additional types of testing, if applicable to your design. If a particular type of testing is not applicable to your project, you must justify why you are not including it.

When writing your testing planning consider a few guidelines:

- Is our testing plan unique to our project? (It should be)
- Are you testing related to all requirements? For requirements you're not testing (e.g., cost related requirements) can you justify their exclusion?
- Is your testing plan comprehensive?
- When should you be testing? (In most cases, it's early and often, not at the end of the project)

5.1 Unit Testing

What units are being tested? How? Tools?

Hardware

LEDs - we will have to test the individual leds that we decide to use to make sure each unit works as expected. Expected LED arrival sometime between Thanksgiving and Christmas.

Microcontroller - Testing the hardware capabilities that are built into the chip to determine if there are better methods for audio transmission (pickup or microphone, wired or wireless) for aiding it displaying the correct LEDs to the user. Testings these different components will allow us to determine if there are more efficient ways to transfer this information or interact with our other hardware.

Sensors - Multiple sensor devices will be tested including pick up, onboard microphone, and onboard accelerometer to detect and measure the audio/vibrational output from the kalimba. Testing will determine the accuracy and speed of sensor pick up and ultimately what strategy we pick to implement

Software

Song Database - test that queries to the database grab the correct data (song and song info)

UI - test that each page is reachable from the homescreen

Song Favorites - test that the user can favorite a song to add it to their favorites list

Uploading a Song - test that a song is playable and the song data is loaded to the database after the appropriate file type is uploaded

Previewing a song - test that a song can be previewable when a user attempts to preview a song from library

User Account Functionality - Unit tests a specific user creating an account, signing in to an account, and verifying that the user has access to account-specific information when navigating throughout the app.

5.2 Interface Testing

What are the interfaces in your design? Discuss how the composition of two or more units (interfaces) are being tested. Tools?

User Interface

- Test that all of the screens in the app are reachable from the homescreen/landing screen.
- We will also need to test that each button or interactable feature produces the intended result.
- Verify that our UI is rendering according to our initial screen sketches produced in Figma (or other UI software).

Physical Interface

- Test unit response time to user input through kalimba tine plucking. Important for sensor and communication timing.
- Once we have decided on all the components that we are going to use, we will also have to test different types of mounting systems to ensure the hardware is attached in an optimal fashion for ease of use.
- LED timing, display organization, and sequences will be tested for user ease of use.

5.3 Integration Testing

What are the critical integration paths in your design? Justification for criticality may come from your requirements. How will they be tested? Tools?

Software Hardware Integration

- This will be testing both using a pickup to physically wire vibration frequency information to a preamp and then on to the software and via an onboard microphone on the microcontroller to send the vibrational information to the software. Tools included in this testing will be LEDs, pickup/microphone, breadboard, power supplies, oscilloscopes, the kalimba itself, microcontroller, software mobile service, and the user themself.

5.4 System Testing

Describe system level testing strategy. What set of unit tests, interface tests, and integration tests suffice for system level testing? This should be closely tied to the requirements. Tools?

End-to-end testing should ensure complete and fast communication between hardware and software systems. System testing would require software-hardware integration, user interface button testing, physical interface input testing, and song database unit testing. System is required to light up LEDs in correspondence with song database order and user input timing while also following commands from the user interface buttons. This would be tested through the combined usage of tools needed in previous testing phases and rigorous user testing.

We would also like to test our system incrementally as we progress through the project. This could start out by initially testing our initial hardware/software interaction as it relates to the system. We then want to continue to test our system as a whole as we progress through the project.

5.5 Regression Testing

How are you ensuring that any new additions do not break the old functionality? What implemented critical features do you need to ensure they do not break? Is it driven by requirements? Tools?

Once we are at the point when we are ready to tie individual units together, we can begin regression testing. We will not have many different units tying together (audio system, lighting system, communication system, and software), but everytime that a new unit is integrated, we will have to test whether everything else still works. For example, on the hardware side, once we can get to the point where the microcontroller is able to recognize specific notes that are played, we will next add the LEDs as visual feedback that specific notes were played. Once the LEDs are added, we will have to ensure that the system is still recognizing the notes correctly.

Using Git we will be able to have many versions of our software available in case a new component broke the previous functionality. We can also implement a CI/CD pipeline to test new software as it is added to our Git repo.

5.6 Acceptance Testing

How will you demonstrate that the design requirements, both functional and non-functional are being met? How would you involve your client in the acceptance testing?

We will demonstrate our design requirements by sticking to the fundamentals that we want to enable a person to use our device on a kalimba and be able to play a song from start to finish, and with enough practice could learn how to play the instrument. We have then been taking decisions that will help us meet that goal by using visuals via LEDs and music notes detection for accurate note recognition to allow us to give the user timely and constructive feedback.

For any decision that strays off of our original plan is to be discussed within each team, hardware or software, to discuss if changing plans or testing different components will result in a better user experience while still following our functional and non-functional requirements. We will involve our client for acceptance testing while we are developing our prototypes and ask them for feedback during major milestones.

5.7 Security Testing (if applicable)

Software:

- make sure that queries to the database are only coming from the predetermined queries necessary to pull the song information
- make sure user/account information is secure in that it is not possible for users to access other users' accounts

Hardware: not applicable, except in maybe a circumstance where a user throws the kalimba to break a window or something, but that's on the user.

5.8 Results

What are the results of your testing? How do they ensure compliance with the requirements? Include figures and tables to explain your testing process better. A summary narrative concluding that your design is as intended is useful.

Ultimately, our product will be able to detect played notes and the application will determine if it is a correct note or not. The software will then send the hardware what note should light up next. This process should be decently easy to test and will largely determine if we have met most of our requirements.

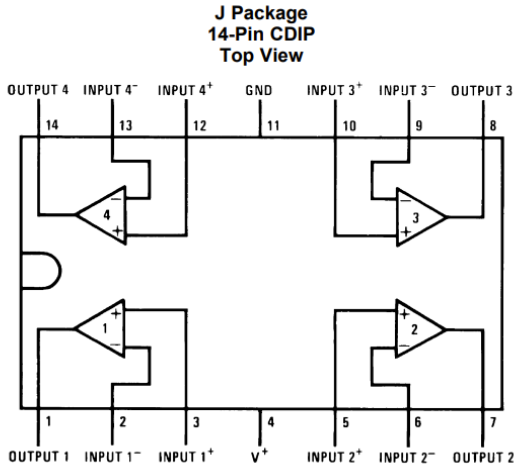
So far two different hardware tests have been conducted. They are both described below.

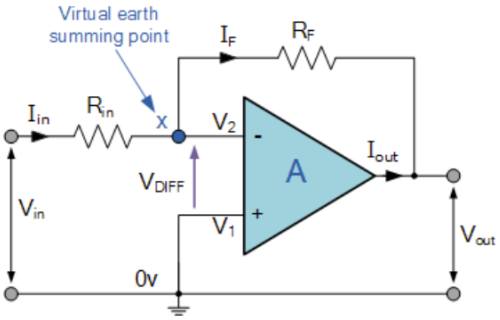
Hardware Test number 1 (10/25)

A pickup for the kalimba has been tested. A piezoelectric pickup was attached to the kalimba and then subsequently plugged into a guitar amplifier. The guitar amplifier has to be turned up to full volume for a soft output sound to be heard; although, the output sound was just as clear and full as the kalimba itself. After some research, it has been determined that a preamplifier is going to have to be added to our design in order for the software to receive a waveform that can actually be used. The issue is that the output impedance of the piezo pickup is high, around 1 Mohm, and the input impedance of most amplifiers is not relatively high, which is why the output on the guitar amp was still soft despite being at max volume.

Hardware test number 2 (11/7)

On 11/7 I tried to measure the voltage signal output of the piezo pickup. To do so, I wrapped wires around the aux cable connected to the pickup (note, I have not tested this aux cable with the guitar amp yet, and I didn't have rubber bands or anything to make sure there was a strong connection between my leads and the aux cable) and interfaced those wires with an LM324 op amp on my bread board, which was set in a typical inverting configuration with an ideal gain of -1E5 V/V. Using a signal from the voltage generator instead of the pickup, I was able to confirm that the amplifying circuit worked as expected . However, even with the 1E5 gain, I was not able to observe a signal on the oscilloscope after plucking a tine. The noise was on the level of 3 to 4 volts peak to peak. Potentially, the signal is still too small to pick apart from the noise even with a gain of 1E5 V/V. Potentially, the connection between the aux cable and the amplifying circuit (haphazardly wrapped wires) was not strong enough to consistently pass signal. Potentially, I have made some other mistake. These are the resources I used to make the circuit so I don't have to look them up later.

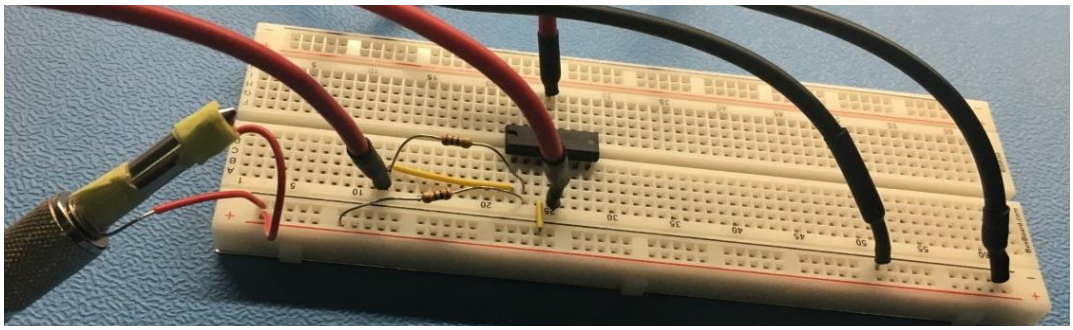


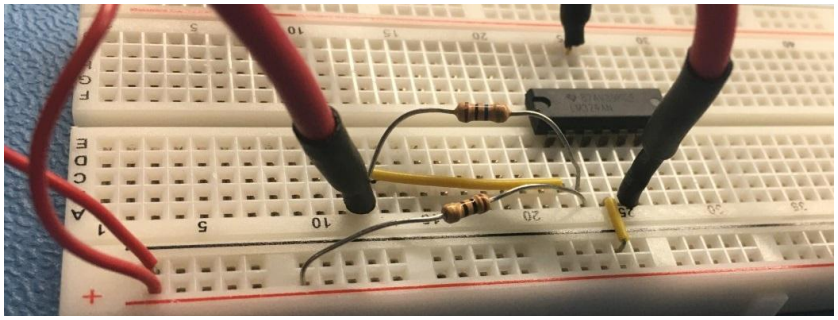


$$\text{Gain (Av)} = \frac{V_{out}}{V_{in}} = -\frac{R_f}{R_{in}}$$

Hardware Test #3 11/9 Update

We set up note detection testing using the audio pickup and an amplifier circuit using an op-amp as pictured below. With an input resistance of 10Ω and a feedback resistance of 10MΩ the amplifying circuit was able to increase the voltage of the input signal from the pick up jack to be accurately measured for note detection.





For the first test, we designed an op amp circuit with the standard LM324 op amp in the non inverting configuration with Rf equal to 2.6 Mohms and Rin equal to 10 Rin, so an ideal gain of 2.6E5 V/V. When plucking the tine that corresponds to C4, we get the following waveform.

C4 should be 262 Hz, and the frequency of the scope reads 258 Hz, only a little out of tune.

When we pluck C4 and C6 simultaneously (frequency of 1047 Hz) we get a resulting frequency of 262 Hz, the frequency of C4, superimposed with the frequency of C6 (1047 hz) just as expected.

Using the multimeter to measure the resistance of pickup, we were getting values that fluctuated from .5 Gohm to 1 Gohm. After a quick google search, I found that multimeter resistance is measured with either constant current or constant voltage. I do not know how the specific multimeter in the TLA was operating, but the resistance of the pickup was fluctuating quite a lot. (over an approximate range of 500 Mohms). This fluctuation is probably due to the fact that the piezoelectric pickup is not technically a conductor, so trying to force current through it or place a voltage across it will not create behavior similar to measuring a resistor. Long story short, the input resistance of the Kalimba is quite high but not known exactly.