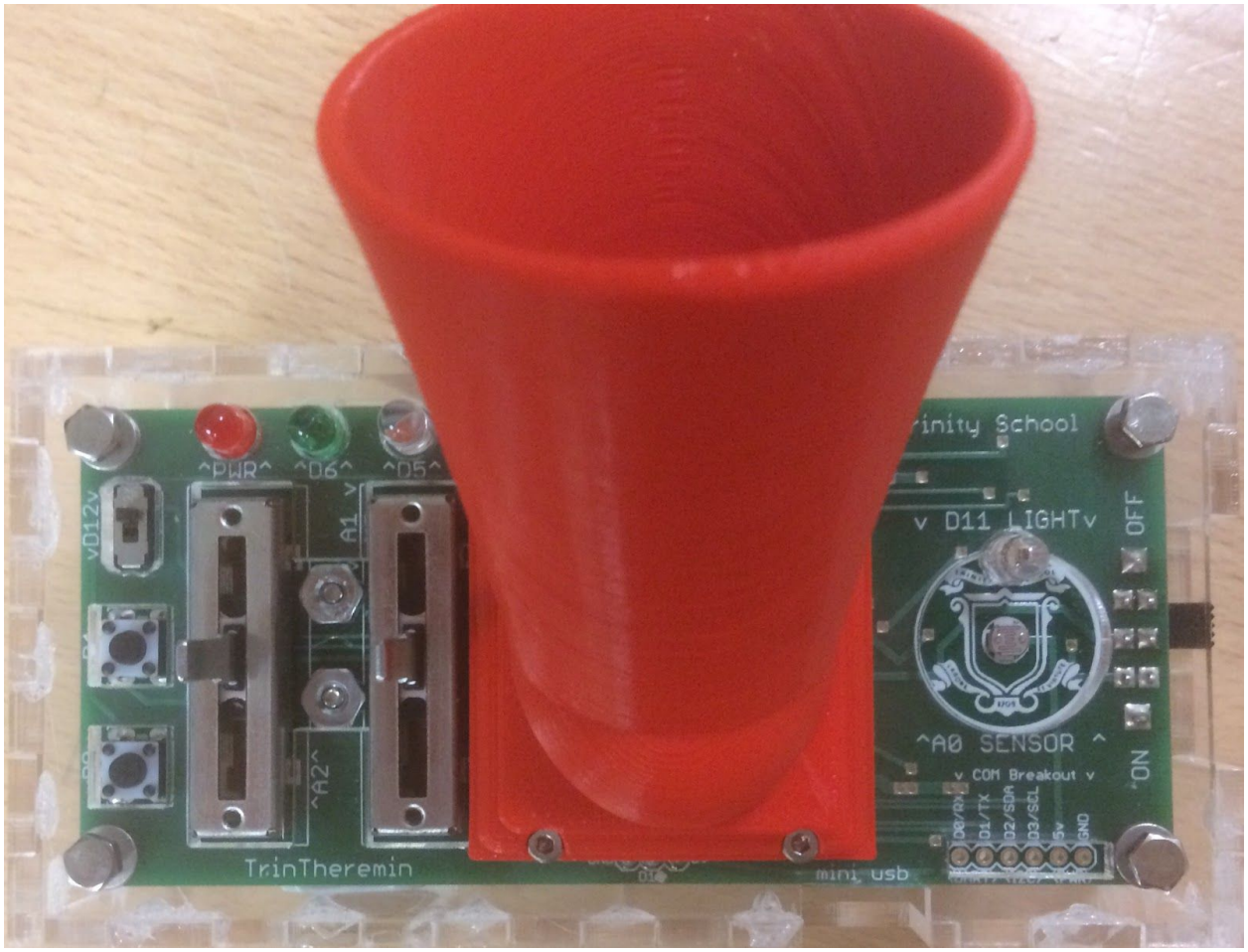


TrinTheremin

Version 1.0

Last modified 12/20/17

Trinity School
Computer Science

Developed by Charles Cunningham in 2017
With help and guidance from Mr. Gohde, Mr. Rose, and Mr. Schober

About

The TrinTheremin is a project inspired by the curriculum of Trinity's second semester CS2 and first semester Advanced CS1 courses. These courses are part of Trinity's four year computer science program. The specific unit that this project grew from challenged students to make an "opto-theremin". A theremin is a musical instrument that can be played without making contact with the device. Originally invented in 1928 by a Soviet inventor named León Theremin, a theremin uses the capacitive properties of the human body to manipulate sound with the distance of the user's hand from the device. An "opto-theremin" is a similar device that uses light instead of capacitance to manipulate sound. It is played very similarly to the theremin, by moving one's hand farther or closer away from the sensor on the device (and thus varying the amount of light exposed to the sensor) to change its pitch.

The TrinTheremin was developed to address a substantial drawback of this unit; unlike software projects, students' opto-theremin projects were physically made on temporary circuit testing and prototyping platforms called "breadboards". After students finished making their circuits, they could keep their code but had to disassemble their projects, essentially throwing away all their hard work. The TrinTheremin is a platform that integrates what is known as a "microcontroller" chip to allow students to upload their code (with a few modifications) onto the device and play the instrument that they designed on a permanent and sound-amplified system. Another benefit of using the TrinTheremin is that it is much more ergonomic than a circuit made on a breadboard and can be played much more easily. Although the circuits designed by students vary greatly, the TrinTheremin has a wide variety of control inputs (sliders, buttons, switches, expansion pins) to cover nearly anything a student might want to include in their design. To translate their code to work with the TrinTheremin students only have to change the addresses of the control inputs used in their old circuit to the ones in the TrinTheremin.

With the help of some of Trinity's circuit design facilities (the "Othermill" circuit board mill, the infrared circuit board oven, and the laser cutter) I was able to produce and test a number of different prototypes of the TrinTheremin until we settled on a final design for version 1.0. Version 2 is in the works, with the possibility of a more efficient amplifier circuit, a headphone jack so multiple students can play the devices without generating too much noise pollution, a battery level sensor so that the circuit can monitor its own battery level and other subtle additions and design changes. See the "Plans for future versions" page for more information on version 2.

Acknowledgements

This project could not have come this far without the help of a few people. I would like to firstly thank Mr. Gohde and Mr. Rose for sponsoring this project and giving help and advice throughout the process. I would like to thank Mr. Schober and Mr. Rose who worked with me to develop a new way of making circuit board stencils with a laser cutter, allowing me to spread solder paste on boards easily and precisely. With the solder paste laid down I could bake the boards in Trinity's modded (for better performance) infrared IC oven to attach surface mounted components. This new technique took a process that normally takes days down to less than an hour, significantly helped the TrinTheremin prototyping process, and will make circuit board production much easier for students in the future. I would like to thank Mr. Gohde for helping to improve the example code and take the project to another level of performance and usability. Furthermore, his constant openness to push and improve projects, rather than just accepting things as "good enough" have really taken this project from a simple hastily put together arduino shield to what it is today. I would like to thank the students of the class of 2019 and current Advanced CS students who have given me invaluable feedback, tips, and ideas. Finally I would like to thank Trinity School for supporting the Computer Science and Robotics programs, without which none of this would have happened.

Example Code Quick Start Guide

1. General Controls and Layout:
    a. The switch on the right side of the board is the power button. If the battery is connected and the switch is in the on position the board will turn on. You can also power the board by connecting a micro USB cable to the USB port. The board will automatically pull power from the USB port if both a battery and a USB cable are connected.
    b. The red light at the top right is the power light (always on when there is power to the board).
    c. The green (D6) and blue (D5) lights will be used in the calibration sequence. In normal playing the green light will be on permanently and the blue light will reflect the pitch of the note that you are playing.
    d. The light sensor that you will wave your hand above to play the device is the round white and gold piece above the Trinity logo on the right side of the board (A0).
    e. The rightmost slider on the left side of the board (A1) controls volume (down is louder).
    f. The left slider (A2) controls the scale of notes you are playing (Major C, Pentatonic, G, etc). There are four different scales loaded onto the board by default and the note that is being played in the scale selected by the slider depends on the amount of light on the sensor.
    g. The switch in the top left corner of the board (D12) turns the audio on and off. The blue light will turn off if you switch the audio off.
    h. The topmost button (D4) will turn on the autoplay feature when pressed. The dancing LED next to the sensor will turn on and the device will play itself, if you follow the autoplay procedure detailed in step #6.
    i. The bottom button (D8) will pause the audio when pressed. If you want to play two notes of the same pitch you can tap this button to separate the notes.
2. Upon turning on the device you need to calibrate it so that it knows the ambient light levels. It starts the calibration sequence immediately, so right after you turn the board on wave your hand up and down slowly and constantly. To set the board's minimums and maximums your hand should go from completely covering

the sensor to exposing it to full ambient light. After about 5 seconds the lights will flicker indicating that the calibration sequence is complete.

3. Finally, now that you are in play mode, the green light (D6) should be illuminated. Turn up the volume and if the device is not making sound, switch the audio on (D12). Once it is playing the highest note in the selected scale will be played when the sensor is exposed to full ambient light and the lowest note will play when your hand is covering the sensor. Vary the height of your hand above the sensor to hit all the notes in the scale.
4. If you hear a clicking sound the volume slider is probably in between two values. Change its position a little bit and this should go away. Two notes can also be blended if your hand is hovering right between them. The device uses a smoothing algorithm to make this blending a little less erratic. As you play the device more you will get the hang of how to avoid and/or take advantage of this characteristic.
5. Finally, you should keep track of battery life. In the next version the circuit will be able to monitor its own battery level but for this one you should just keep a rough estimate of how long you have played it. A fresh 1200mah battery will last 3-6 hours of cumulative audio output. 3 hours at full volume, 6 hours at half volume, much longer at low volumes. If you start to approach these values and the device starts behaving abnormally the battery should be changed. The battery is kept in the metal clip on the bottom of the board.
6. The autoplay feature of the TrinTheremin is designed to work best with lower ambient light levels. If the autoplay light is turned on in a bright room, the dancing LED will not make much of a brightness difference to the sensor, and the autoplay feature will not work well. Therefore, it is recommended that you put your hand an inch or so above the dancing LED and sensor on the Trinity logo before turning on the autoplay feature. The exact height of your hand above the sensor does not matter as the code will sense the ambient light conditions with your hand over the device and account for it appropriately. When you turn it on it runs through a very brief (unnoticeable) calibration procedure that senses the ambient light levels with your hand over the sensor. Every time you turn on the autoplay feature it will calibrate to wherever your hand is, but will not adapt if you move your hand while the autoplay feature is running (while you are holding down the autoplay button).

Expanded Documentation

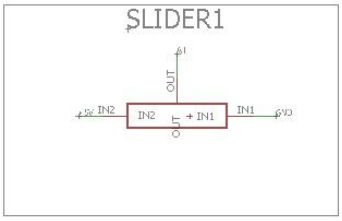
About the Circuit

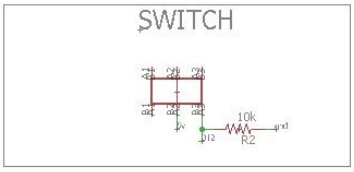
TrinTheremin, like the Sparkfun Digital Sandboxes used in CS1/CS2, is an all in one programmable circuit that allows the user to process/control different inputs/outputs on the board. Its ATMEGA microcontroller chip has its own CPU (central processing unit; the brain of the device), flash memory (where your program is stored), SRAM (where the sketch creates and manipulates variables while running), and EEPROM (NON-VOLATILE storage space programmers can access/use; EEPROM will keep its stored data even without electrical power (when the unit is off), as opposed to volatile memory (like SRAM) that would lose its data without electrical power). While you probably won't need to use any EEPROM for this project (although feel free to do so if you have a use for it), you will constantly be using the CPU and SRAM, as well as the flash memory (where your program lives). See this chart for how much memory the microcontroller on the TrinTheremin (the ATMEGA 32u4) has of each main type (there are other memory types onboard that I won't go into):

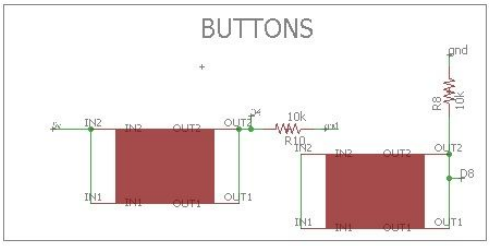
Memory Type:	Amount of storage (bytes):
Flash	32000
SRAM	2500
EEPROM	1000

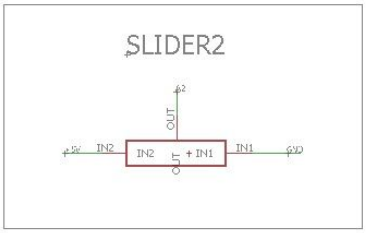
The flash is not all available, however, as the board's "bootloader" (the software that allows you to upload Arduino code) takes up about 4kB, or 4000 bytes, leaving about 28kB usable for storing code. The bootloader is already installed on all the boards.

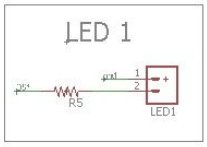
Beyond the chip (and its dedicated hardware such as its crystal and resistors and capacitors), all the circuits on this board should be pretty familiar (with the exception of the speaker's circuit). See the circuit design pictures folder. On the left side of the schematic is all the dedicated Arduino hardware, however the right side shows the circuits for all the peripherals (the buttons, switch, light sensor, LEDs, etc). Take a look at the right side to see how simple (and common) the circuits behind the peripherals are. The circuit for for the switch, for instance, just includes the switch and a pulldown resistor (which keeps the voltage at 0V unless the switch is closed to 5V) and is connected to a digital pin (D12):

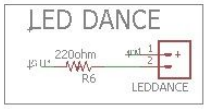


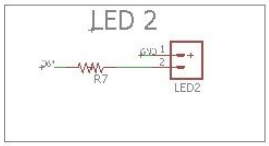


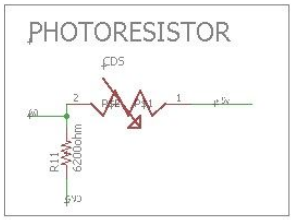


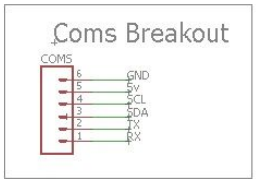


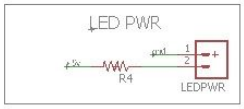












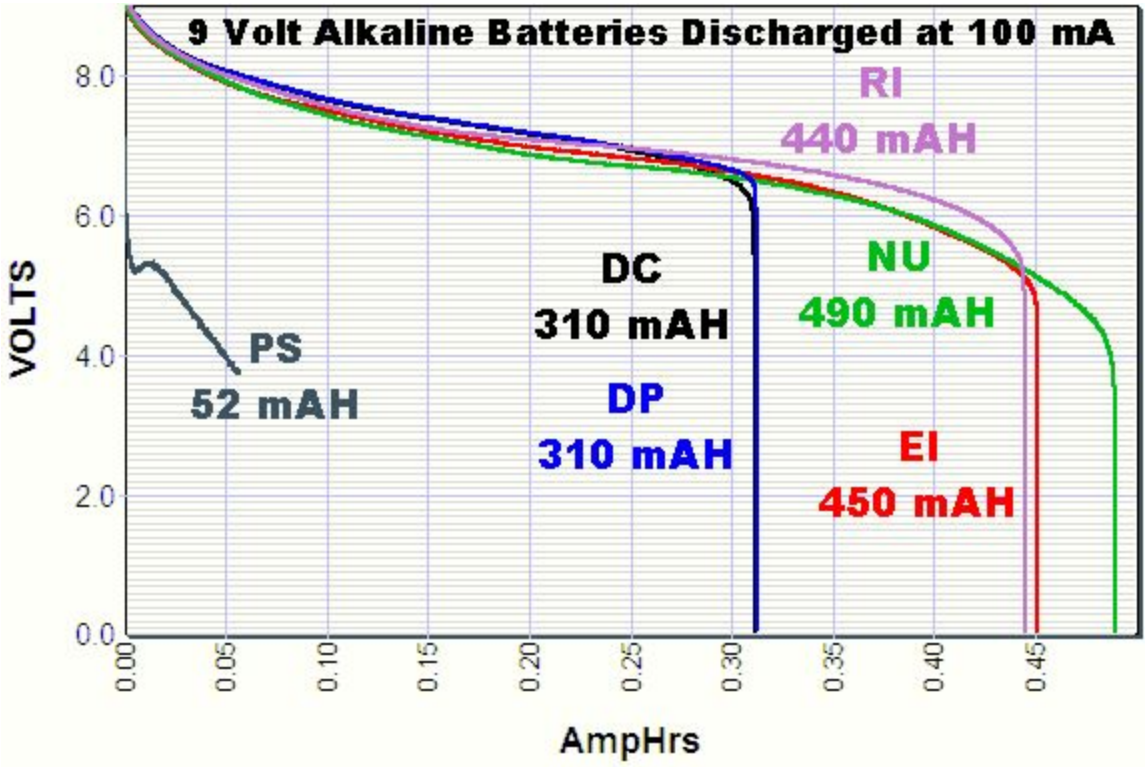
Usage

The TrinTheremin was purpose built for Trinity's CS2 and Advanced CS1 programs (specifically the physical computing and digital logic units, respectively), although it could be used in a wide variety of applications that use inputs and outputs involving sound and light.

Battery

The TrinTheremin's battery should be connected to the 9v battery connector on the bottom of the board and stored in the clip under the slider (also on the bottom of the board).

Like any device, the TrinTheremin will eventually run out of battery. Trinity's collection of Apex batteries have a 595mAh capacity. Milliamp hours (mAh) are a unit of electrical charge (one milliamp hour is about 3.6 coulombs, a unit commonly taught in Trinity physics that is generally used when working with static electricity rather than circuits). This means that the batteries can supply 595 mA of current for one hour. The TrinTheremin draws about 400 mA when the speaker is on at full volume so at minimum the TrinTheremin will run for about 1.5 hours (with everything running and its speaker on at full volume) continuously. If you do not have the speaker on the device draws only about 25 mA, which will yield about a day of continuous use (so do not feel the need to turn off the device while programming it as having it on does not use up much battery life) To more accurately determine how much battery life the device has left (and without keeping track of how much time you have used it for), you can measure the remaining battery life with a multimeter (located in the top drawer in the drawers by the door in the computer science lab -- ask Mr. Rose or Mr. Gohde if you are unsure where this is) and a few quick calculations.



First, set the multimeter to measure DC voltage and make sure the leads are connected to the right ports (red to positive (+) and black to negative (-, sometimes referred to as COM) -- see this tutorial on multimeters. Then measure the voltage of the battery (touching the red lead of the multimeter to the positive terminal of the battery (marked by the + symbol on the battery) and the black lead to the negative terminal (marked by the -). Here is a chart that shows the battery life estimates for an alkaline cell:

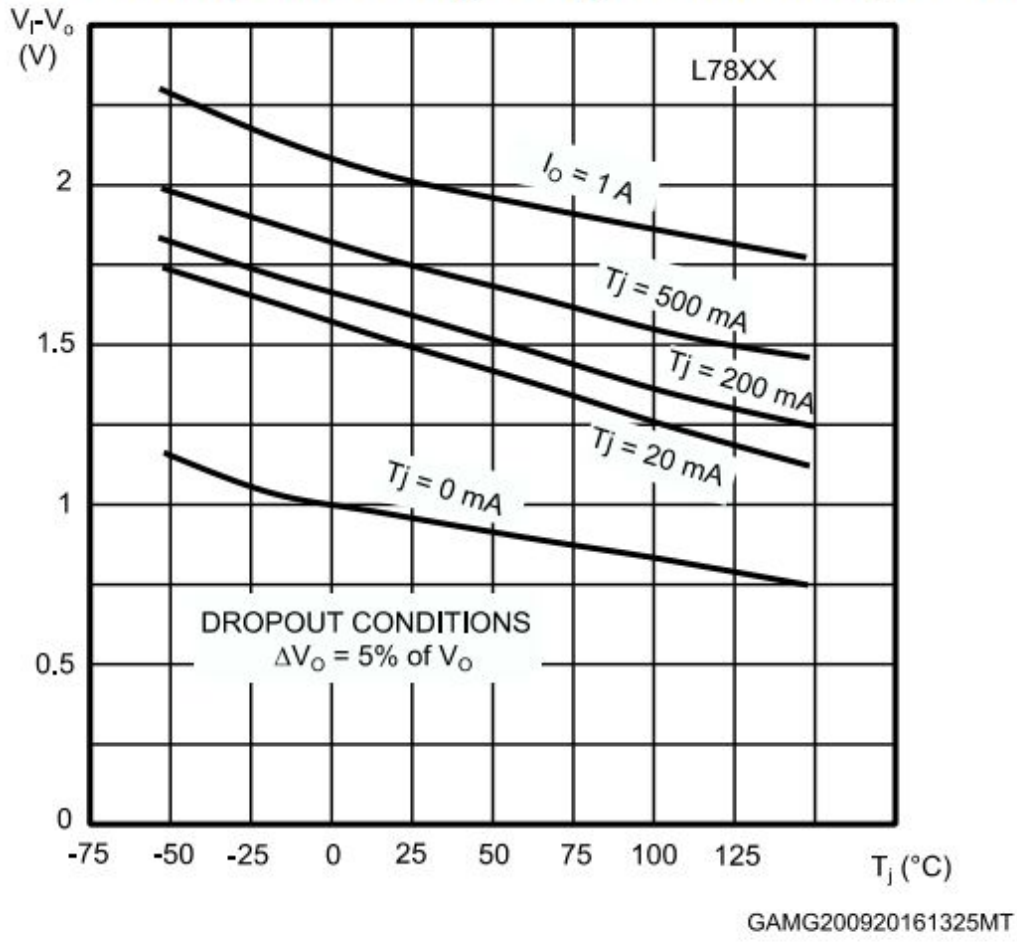
Alkaline Voltage Level and Range (V):	**Usability:**
1.50	New and unused
1.35 – 1.49	Good
1.20 – 1.34	Average
1.10 – 1.19	Low
0.90 – 1.09	Very Low
< 0.90	Dead – Discard!

Given that alkaline 9v batteries (Trinity's Apex 9v batteries are alkaline) are 6 alkaline cells, we can multiply through these ranges to find voltages that apply to the reading from the 9v battery:

9v Battery Voltage Level and Range (V):	Usability:
9	New and unused
8.1 – 8.94	Good
7.2 – 8.04	Average
6.6 – 7.14	Low (Unusable below about 6.75v)
5.4 – 6.54	Very Low (Unusable)
< 5.4	Dead – Discard! (Unusable)

There is one more factor to consider, though, and that is that the TrinTheremin needs about 6.75v to run properly. This is because the voltage regulator (a chip on the board that takes the voltage being supplied by the battery and regulates it at a constant 5v for the circuit to use) needs at least 6.75v to be able to output 5v as it's supposed to. This extra 1.75 volts it needs is called the "dropout voltage" -- this chart from the chip's datasheet was how I found that the dropout voltage is 1.75 volts (assuming 25C temperature and 500mA current draw):

Figure 28: Dropout voltage vs junction temperature



If the board is acting weird or blatantly executing programming commands incorrectly (that's not because of mistakes in the code) it is almost certainly because its does not have the necessary voltage (so measure it and replace the battery if necessary). There is a small range of voltages where the board will turn on but may work improperly.

Programming

The TrinTheremin has the same chip as an Arduino Micro, so it will behave almost identically to that board. Important caveats to note:

- When using Serial (rx/tx) on the "COM breakout" you must use Serial1 instead of Serial. Since this chip has its own embedded usb circuit, Serial.print will print to the Arduino console as usual, however Serial1 will reference the hardware serial ports (rx/tx). So you would use Serial1.begin(9600), Serial1.print("text"), etc to use the HARDWARE serial. On an Arduino Uno, for example, which has an external USB circuit that is connected inline with the hardware serial ports, the Serial command references both the hardware and the usb serial (so Serial.print will output serial on the tx port but also print to the Arduino developer's console)
- Because the chip has built in USB, when you plug it into your computer it will be recognized as a mouse or keyboard (and you can actually use it as such -- more on that here).
- When you upload your code, select "Arduino Micro" under boards, and the appropriate port under ports.

Example Code Notes

- When programming the slider to control the volume of the speaker, map your analog reading from 0-1000 to 0-10. If you use 0-1023 the full volume tones will sound distorted because it will flicker between volume level 9 and 10.
- The calibration procedure in the example code is this: i. When one LED is blinking do not cover the sensor at all -- it is measuring the ambient light levels; ii. Once the speaker makes a noise and the LED stops blinking cover the sensor with your hand iii. Once both LEDs stop blinking and then light up in a pattern the calibration process is complete
- The autoplay feature can work very well if you put your hand over the dancing LED and the sensor. It will work fine if you leave it in ambient light, but the more ambient light there is, the worse it will work (this cannot be fixed in calibration -- think about it like the LED's light is getting drowned out).
- The toneAc library is used to control the speaker. To download toneAc see this link. Here is the toneAc reference.

Plans for future versions

If any future versions of the TrinTheremin are produced these are some of the goals that did not get included in this initial version:

- *A more efficient amplifier* -- the amplifier circuit on the TrinTheremin right now is very simple to use and understand (both in software and in hardware), works with the ToneAC library, and has pretty good quality sound (meeting the goals of the board). However in future versions a more sophisticated amplifier circuit might be combined with a possible TrinTheremin Arduino library to keep the software simplicity of ToneAC while improving quality and GREATLY improving efficiency (battery life would be improved immensely).
- *A voltage divider circuit* -- this is one of the easiest (and also most helpful) changes that could be introduced in future versions. Essentially this would take the voltage of the battery, divide it in two (9v becomes 4.5, 7 becomes 3.5, etc), and feed it to an analog input (0v-5v yields a 0-1023 reading) on the microcontroller. This would allow the programmer to measure the voltage of the battery. With this they could create an integrated battery life check that would alert the user if the battery were low.
- *Headphone jack* -- a headphone jack could be added to the board in addition to the speaker so the user could play with headphones on. The port would have a switch in it that would be connected to a digital pin so the programmer could automatically turn off the speaker when headphones are plugged in.
- *Reduced cost* -- right now the TrinTheremin costs about 25$ per board to produce, however with hardware changes and different production methods, this could potentially be significantly reduced. For reference an Arduino Micro costs about 22$ (with no inputs or outputs built in)
- *Expanded documentation and a more seamless user interface -- maybe a website?* -- Right now trintherem.in redirects to this github repository, however in the future a website dedicated to the TrinTheremin could be produced.