Brian M^cLaughlin

EMID Project 2 Report

7 May 2014

Piezo Kalimba

Design Goals

The initial objective of this project was to design and build an expressive handheld electronic instrument that is modelled after the kalimba, or thumb piano. We wanted to implement the electronics and software knowledge we gleaned from the Rubber Banjo in a new, more elaborate system. We also wanted to put a greater emphasis on the musicality of the instrument, including synthesizer sounds and playability.

General Overview

The Piezo Kalimba is a MIDI controller which interfaces with an Arduino Uno to control a set of sounds in Reason through a Max/MSP patch. The structure of the device is modelled after a traditional Kalimba: eight metal keys are the focal point of the handheld instrument. **Figure 1** shows the piezo sensors attached to each key that are triggered when a key is plucked. The player holds two PVC pipes that suspend the kalimba. On each of these PVC pipes are two force-sensing resistors and a selector switch, as shown in **Figure 2**. There is also an RGB LED mounted to the top so the performer and the audience can better visualize what is happening on the instrument.

Materials

- 1 Arduino Uno Microcontroller
- 8 MiniSense 100 Piezo Sensors

- 4 Force-Sensing Resistors (0.5-inch diameter)
- 1 Fender-Style 5-Way Guitar Selector Switch
- 1 Gibson-Style 3-Way Guitar Selector Switch
- 1 Radioshack Universal Component PCB with 780 Holes
- 6 TLC272 Dual Single-Supply Op-Amps
- 8 1N4004 Rectifier Diodes
- Resistors: 1 MΩ (x16), 10 kΩ (x4), 22 kΩ (x8), 1 kΩ (x10)
- Capacitors: 0.1 µF (x8), 0.01 µF (x8)
- Wires: Red (+5V), Black (GND), Orange (analog), White (digital)
- 84 Header Pins
- Rainbow Jumper Cables
- Wood Platform
- PVC Pipes
- Wooden Dowel
- Thick Metal Wire
- 8 Kalimba Keys
- Neoprene Foam

Hardware

To condition the signal coming from the piezo sensors, I implemented the same circuit I used for the first project, multiplied by eight. The signal from the piezo is rectified with a diode, passed through an envelope-detector RC filter into a non-inverting op-amp configuration with some capacitance in parallel to the feedback resistor to damp chip oscillation. This signal is sent to the Arduino analog pins. To condition the FSR signal, I implemented a simple voltage divider configuration with an impedance buffer. The output was also sent to the Arduino analog pins. All digital signals were sent to the Arduino digital pins. Buttons were implemented by acting as a short between the 5V source and the 1 kΩ pull-down resistor. The two selector switches worked in the same manner, and the output from each acts in a sense as a digital logic selector. The RGB LED took three signals from the PWM Arduino pins and sent each through one of the three colored LEDs with a pull-down resistor to ground. This allowed the intensity of each color to be controlled by the duty cycle of the pulse-width modulation. The schematic for the system analog and digital circuitry can be found in the appendix of this report.

Enclosure

The body of the instrument was made of a plywood panel. Two PVC pipes were used as handles for the player to both hold the instrument and control the FSRs and switches. The keys were mounted in a similar way to a traditional kalimba, a wooden dowel is screwed into the body to hold the piezos in place (in-between the dowel and two metal wires on top of the body). Neoprene foam, hot glue, and electrical tape were used on the body of the kalimba and on the keys to damp the pluck vibration noise transferred from one key to another. The Arduino and electronics board were mounted to the back of the wooden panel. Neoprene foam also covered the FSRs themselves for a smoother feel. A shot of the instrument as a whole can be seen in **Figure 3**.

Software

The peak detection algorithm for the piezos was similar to the previous project, but more consideration had to be made for the vibration noise transferred from one key to another, which involved raising the noise floor in software, thus limiting the dynamic range. To combat the noise issue, only one note per sensor was allowed to sound within a certain time frame (on the scale of tens of milliseconds). So if the vibration from one piezo transferred to the others, only the first piezo would sound. A sleek user interface allowed for more natural interaction between human and machine, giving the player the vital musical information visually. We tried to maximize the functionality of each physical controller in software, thus keeping the playing intuitive and the instrument simple and compact. The Fender-style switch set the musical scale playable by the piezo when the Gibson-style switch is either up or down. If the Gibson switch is centered, the Fender switch sets note-duration, from very short to infinite sustain (this setting allowed LFO and filter changes more room to breathe). To solve the latching problem with the FSRs (how to determine which value is the 'final' value), two FSRs are used for each parameter. One increases the parameter during the positive part of the output slope and the other decreases the parameter during the positive part of the output slope. The parameter is determined by the position of the Gibson switch. For one switch position, two FSRs are dedicated to filter frequency while the other two are dedicated to filter resonance. For another switch position, two FSRs are dedicated to LFO rate and the other two are dedicated to LFO amount. For the middle position, the FSRs are dedicated to wet/dry mix for reverb and delay. The two push-buttons offset the scale by one note, so any note could be played using the eight kalimba keys.

Sounds

Reason software reads in the MIDI output from Max/MSP to control sample sets and synth patches. We mostly used one Subtractor patch in conjunction with reverb and delay units, but also used a piano sample set via NN-19, mainly for note clarity in diagnostic tests. We designed a Subtractor synth patch that was polyphonic and harmonically rich, so that changes in filter frequency and resonance were especially noticeable. The physical controllers were mapped to the Reason software controllers as described in the software portion of this report.

Bumps in the Road

The biggest hurdle was dealing with pluck vibration transferring throughout the body of the instrument. We were interested in using piezos due to their extreme sensitivity to mechanical vibrations; however this sensitivity made our instrument susceptible to inter-channel mechanical interference. We solved the problem with heavy mechanical damping and software thresholding, but if we had realized the mechanical flaw in using traditional kalimba key-mounting, we would have spent more time trying to isolate each piezo sensor's motion from all the other piezos, thus greatly reducing the physical noise of the system and expanding the possibilities by not having to limit the range in dynamics and speed through software. Also due to the intense amount of soldering involved, and the fact that a single circuit was duplicated many times by hand, it was practically inevitable that something wouldn't turn out quite right. Luckily the only soldering bug occurred in one spot: one of the piezo channels. Since the circuitry was so dense, it would have taken a very long time to debug (although I'm beginning to think I may have fried a transistor or two in one of the op-amp chips while soldering when I made some small fixes with the op-amps still in the dip sockets). Given the magnitude of the circuitry involved, this is a very small hand-soldering bug that could easily be fixed or avoided by laying out the circuit on a printed circuit board with surface-mountable components.

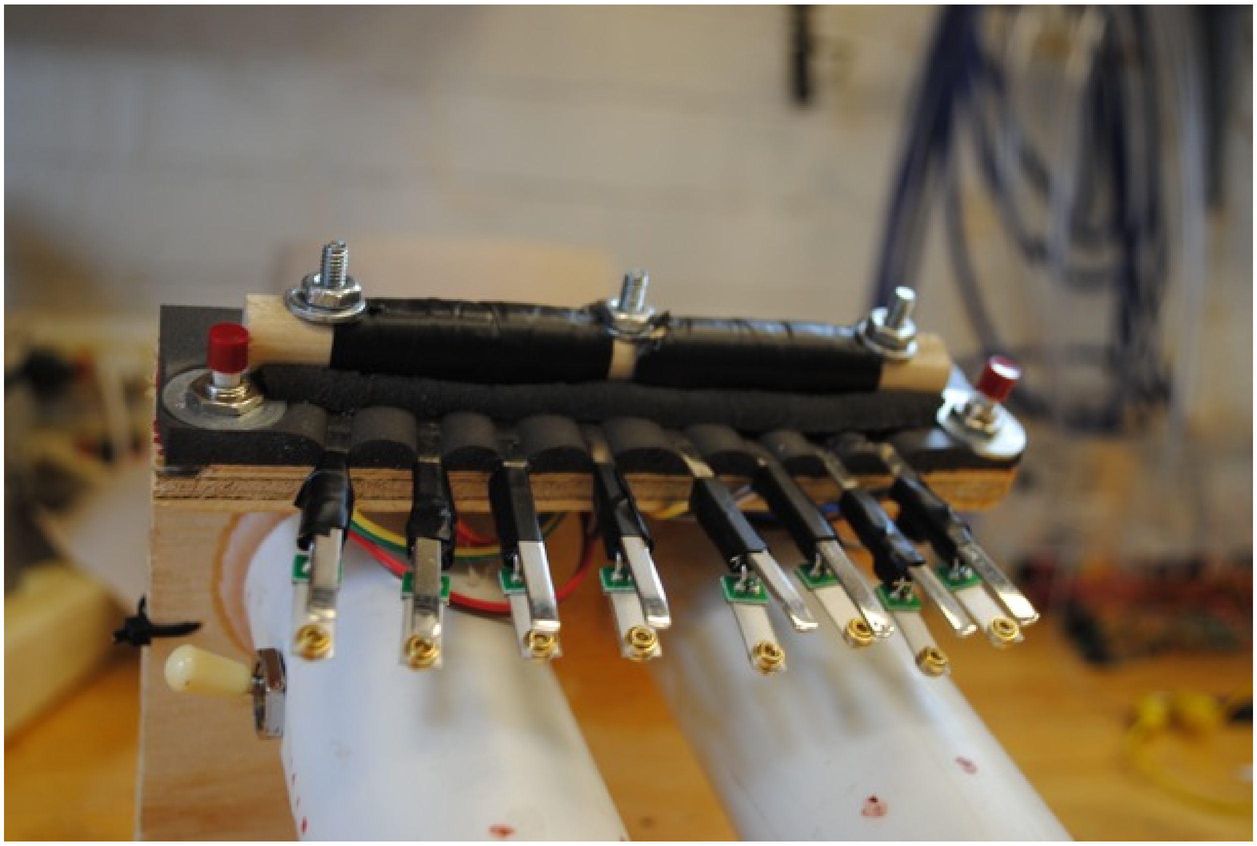


Figure 1: Piezo sensors mounted to kalimba keys

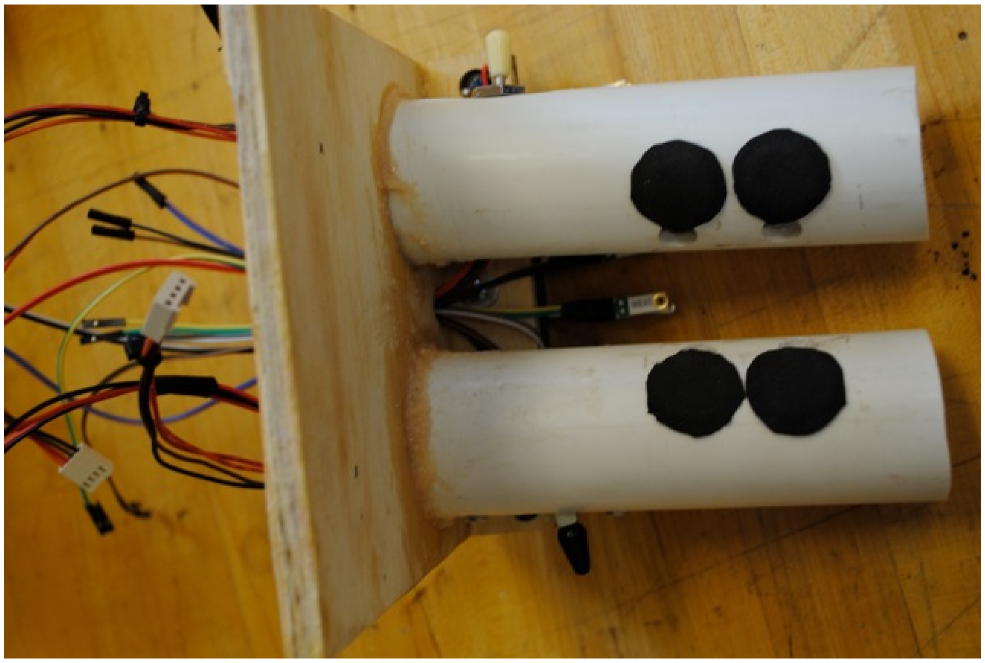


Figure 2: FSRs mounted to PVC pipes with neoprene pads

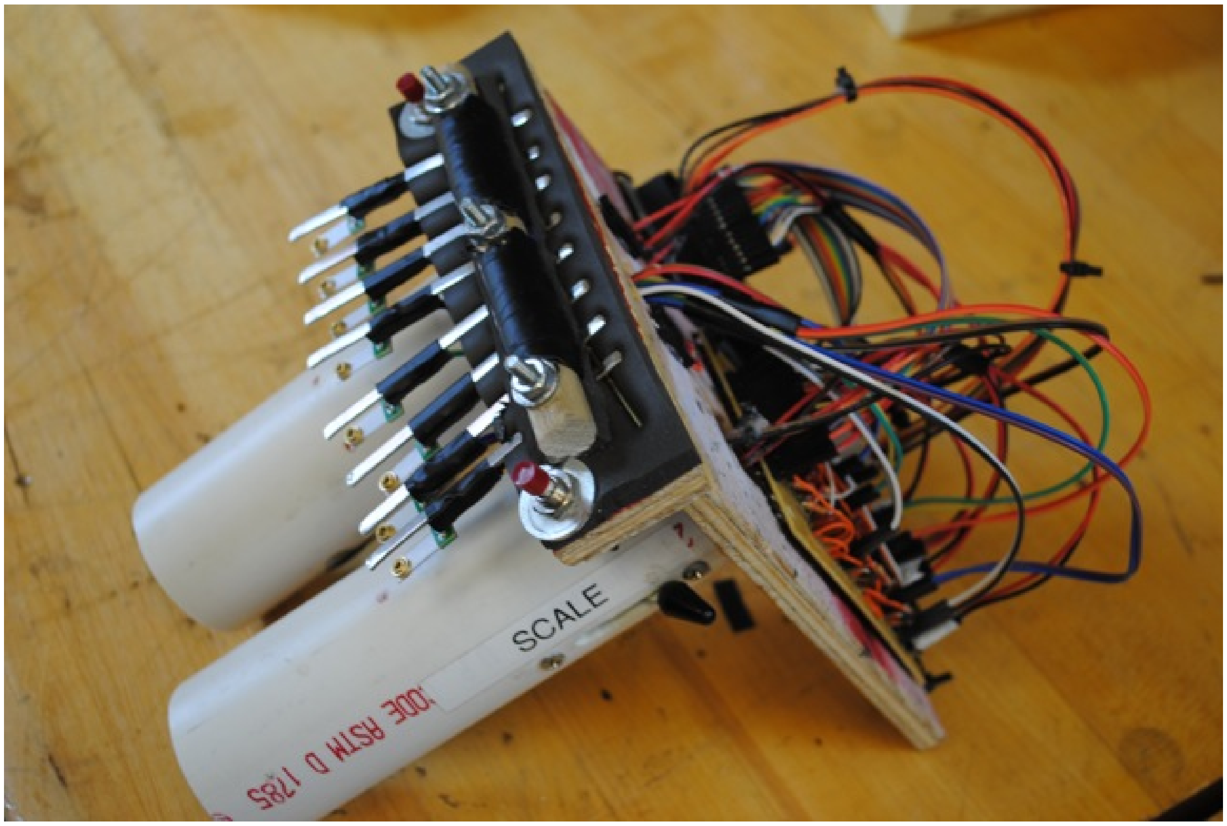
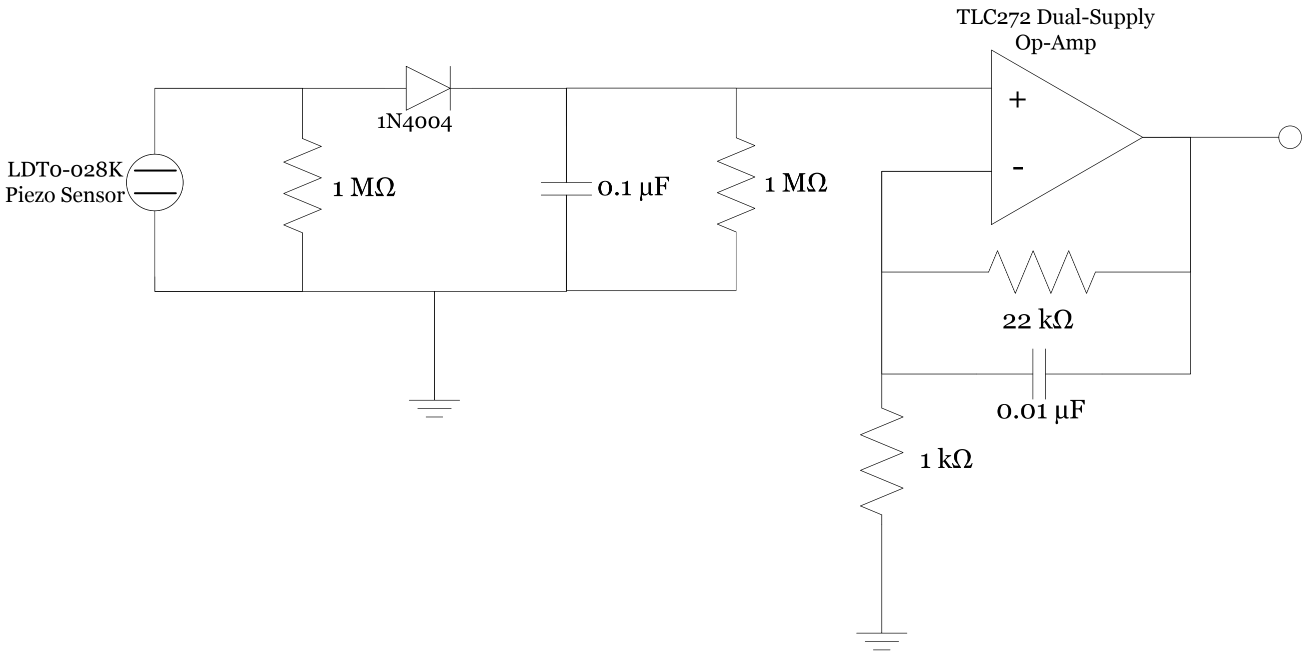


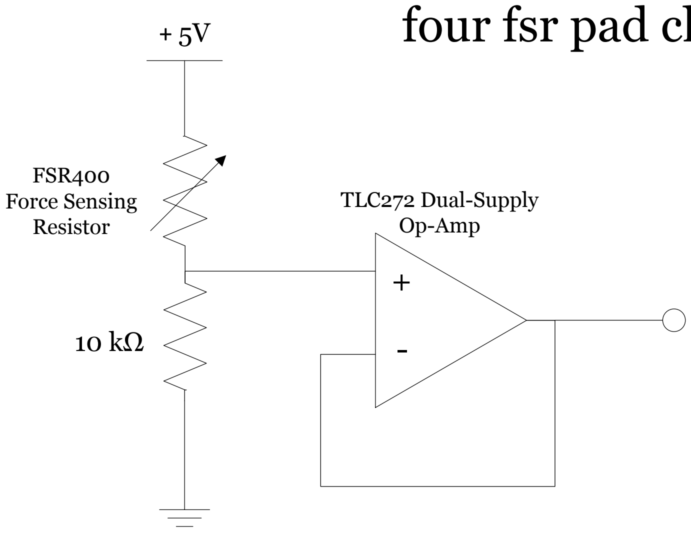
Figure 3: System overview

ES 95 Project 2

eight key-detection channels



four fsr pad channels

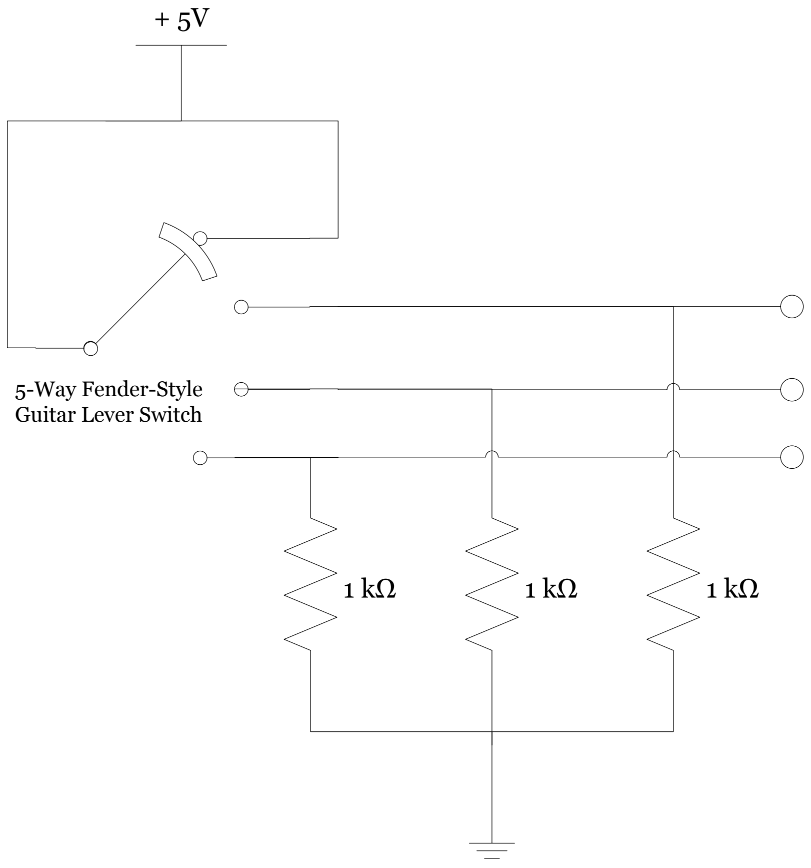


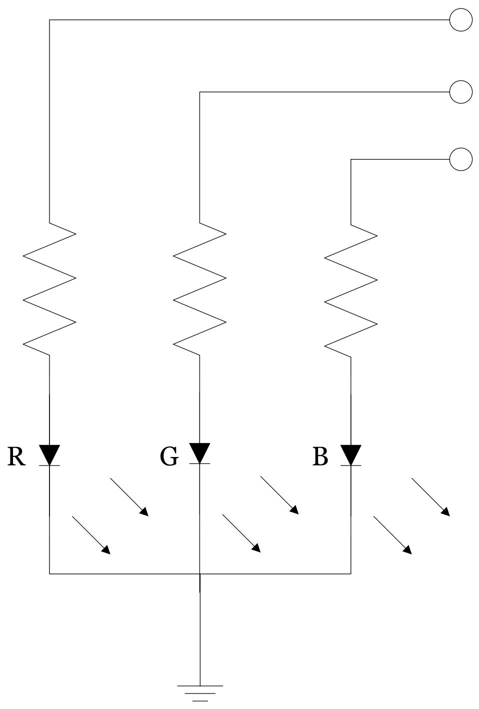
Piezo Kalimba

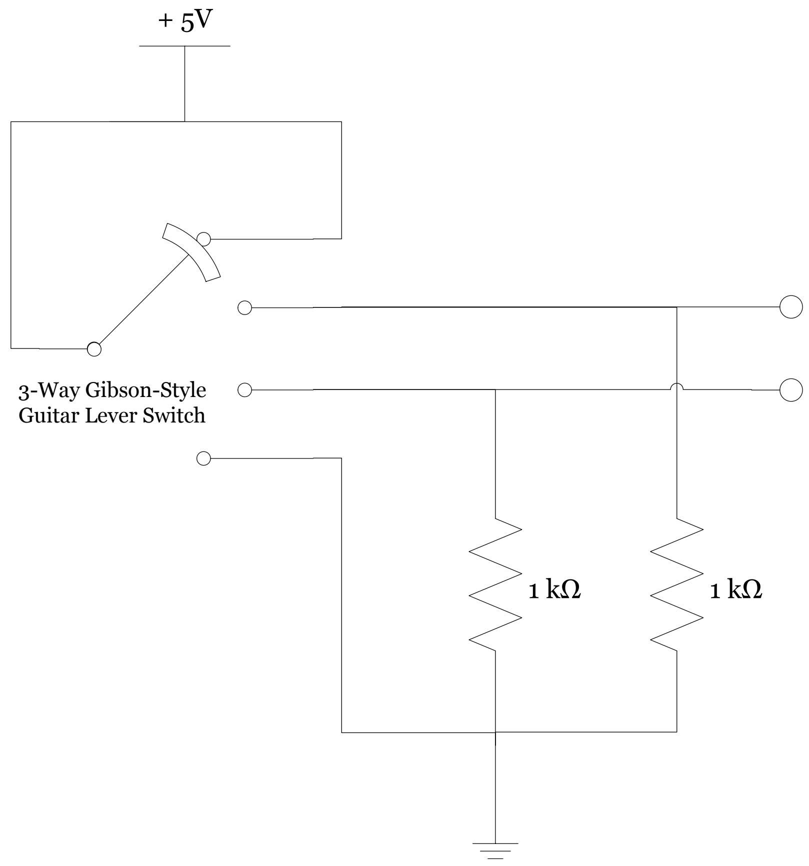
Circuit Diagram

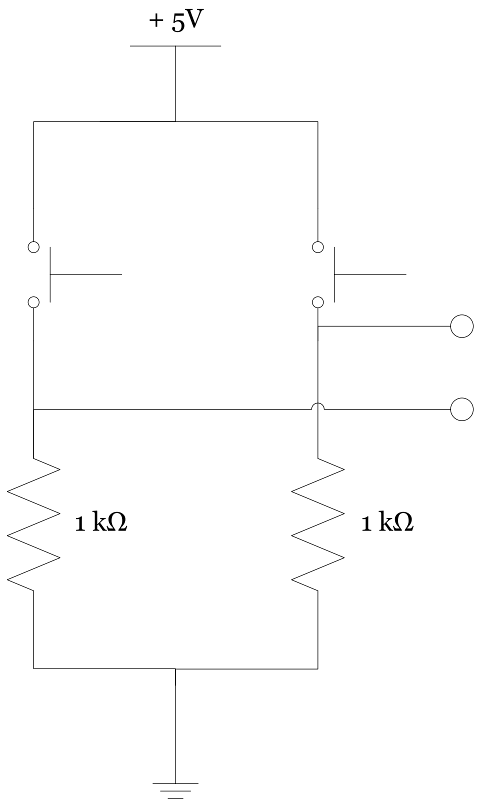
for Arduino analog pins

ES 95 Project 2









Rubber Banjo

Circuit Diagram

for Arduino digital pins