SSSSS: Seventies-Style Sight and Sound System

Jonathan Lam (lam12@cooper.edu)
Steven Lee (lee70@cooper.edu)

The Cooper Union for the Advancement of Science and Art
ECE394 Electrical and Computer Engineering Projects II
Prof. Stuart Kirtman

May 14, 2021

1 Abstract

The color organ is an electronic speaker for music that has colored lights corresponding that light up based on the the frequency of an audio signal. The goal of this project is to implement a simple color organ with three frequency ranges: bass (225-450Hz), midrange (1000-1500Hz), and treble (4800-5500Hz).

2 Introduction

As described in the abstract, a color organ is a device that can detect different frequencies and light up different LEDs based on its input.

Color organ, a device that allows people to enjoy both music and make an relation with color was a creative invention during the 18th century for people to enjoy music[1]. Even though it was all manual at the time, it was a big hit and people loved associating the two together. Fast forward to the late 1960s, color organ showed up as a different form, with the ability to detect different frequency/amplitude to show different colors and making it a lot more easier to operate.

3 Block diagram

The block diagram is shown in Figure 1. The project comprised two major components: amplifying the signal enough to power an $8\,\Omega$ speaker, and filtering the signal into the respective bands. These will be described later in greater detail.

4 Schematic diagrams

4.1 Filter design

We have constructed three active bandpass filters for the design, where each one would be detecting a different frequency. The formula we use to determine this is:

$$\omega_c = \frac{1}{2\pi f c} \tag{1}$$

The schematics for each filter are shown in Figure 2.

[1] `https://en.wikipedia.org/wiki/Color_organ`

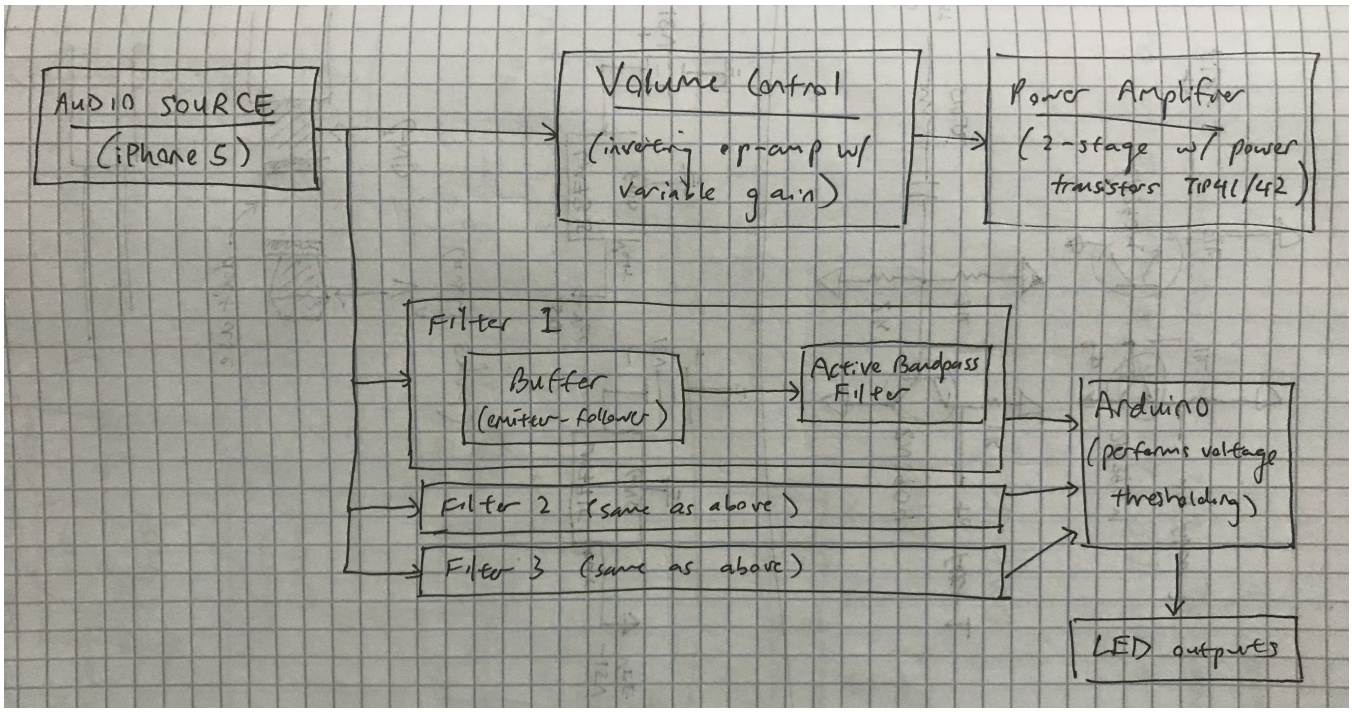


Figure 1: Block diagram of the overall color organ

4.2 Volume control and power amplification

For volume control, we use a simple inverting amplifier, where the feedback resistance is a 50 kΩ varistor. The gain for this is:

$$A_V = -\frac{R_{vol}}{R_{in}} \tag{2}$$

In the case of $R_{vol} = R_{in}$, this allows us to decrease the amplitude already. To allow for amplification, then R_{vol} should be greater than R_{in}.

To be able to drive the speaker, we need a power amplifier. Originally, we tried a single op-amp (LM741) amplifier to power the speaker, but this only resulted in noise at the output – clearly, the output of this ordinary op-amp is not powerful enough to drive the speaker. Similarly, a single BJT (2N3904) was not enough.

During the labs we discovered the Darlington pair topology, in which two NPN transistors are chained in a way such that their current gains are multiplied. This allows for very high current gains. In our kit we have Darlington pair power transistors TIP120 (NPN) and TIP125 (PNP). However, we were unable to find out how to implement this correctly in the circuit, due to a lack of examples online and a lack of experience with setting up Darlington pairs or power transistors.

We were able to use a topology similar to the Darlington pair called the Sziklai Pair (also known as the "Complementary Darlington") which uses a NPN and PNP. They offer several advantages over Darlington pairs, which are noted for example on the Wikipedia page[2]. Our reference for the schematic is from "DIY Amplifier using TIP41C and TIP42C transistors"[3]. This example used a 12V source and ground as its power rails (note that the VEE rail is not used for this subcircuit). We kept most of the same values except that we used the 15V from VCC, used a $20\,\mathrm{k}\Omega$ rather than a $18\,\mathrm{k}\Omega$ collector resistor, and made some minor adjustments to how the transistors were connected to VCC. The final schematic is shown in Figure 3.

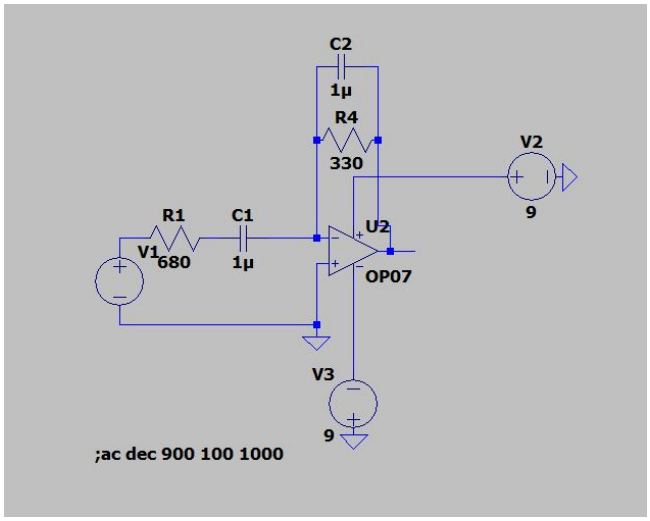
The result sounds fairly good, except that it causes the TIP42, the LM7815 (VCC voltage regulator), and the speaker to heat up. (At first, we did not know this, and it seems to have damaged one of our speakers so that the output sounds distorted and fainter.) As a result, the TIP42 and LM7815 were "mounted" onto a heatsink (via tape, as the solder didn't stick onto the copper heatsink without flux, which we didn't have on hand). This solved the heating issue for the chips, but the speaker still heated up very quickly, thus limiting us in how long we could run the color organ continuously. We couldn't figure out the exact computations to set up the power correctly, but this is something to be investigated for a practical amplifier circuit that would be run continuously for any reasonable period of time. More power efficient speakers (e.g., Class D speakers) are much more complicated and have more stages in order to achieve higher efficiency; our two-stage filter is probably loosely a Class A amplifier[4].

4.3 Power regulation

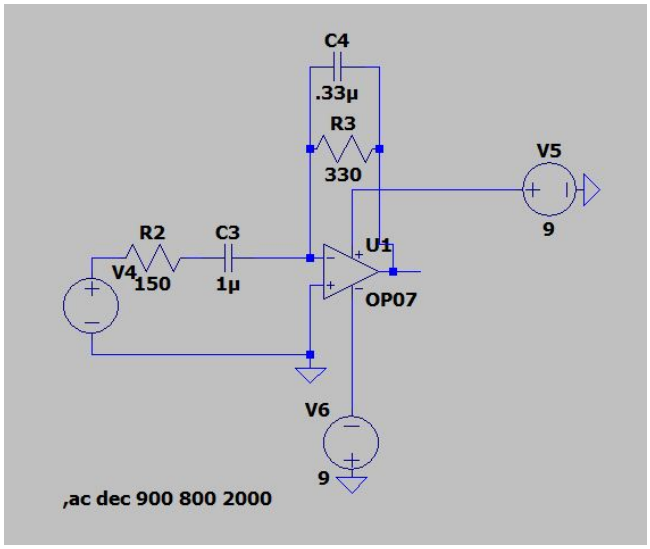
The VCC and VEE power rails were $\pm 15\,\mathrm{V}$. These each used two 9V batteries and the appropriate voltage regulator (LM7815 and LM7915, respectively). See Figure 4.

We had a few problems with the power rails. Firstly, there was a small periodic noise in the VEE rail, even when nothing but the voltage regulator was plugged into it. We smoothed this out with a decoupling capacitor connected between VEE and GND. Another issue was that the VCC rail dropped significantly when powering the speaker and power amplifier circuit – this was due to drained batteries and was fixed when the batteries were replaced.

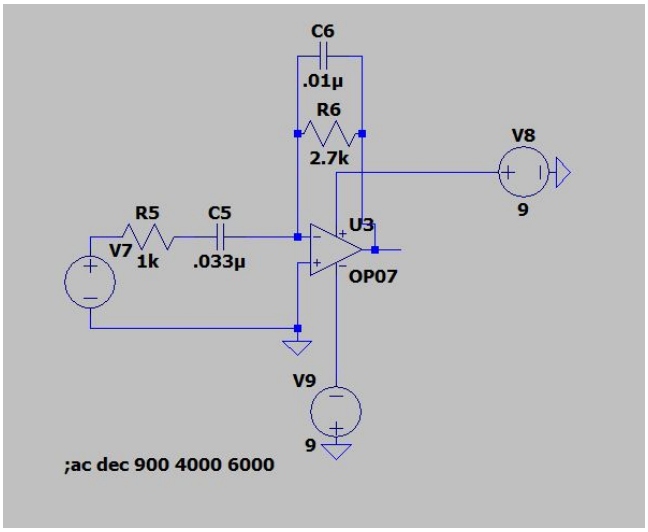
[2] `https://en.wikipedia.org/wiki/Sziklai_pair#Advantages`
[3] https://www.youtube.com/watch?v=jFf6VnkpF6Q
[4] `https://en.wikipedia.org/wiki/Power_amplifier_classes`



(a) Bass Filter: R1 = 680 Ω, C1 = 1 µF , R2 = 330 Ω, C2 = 1 µF



(b) Midrange Filter: R1 = 150 Ω, C1 = 1 µF , R2 = 330 Ω, C2 = 0.33 µF



(c) Treble Filter: R1 = 1 kΩ, C1 = 0.033 µF , R2 = 2.7 kΩ, C2 = 0.01 µF

Figure 2: Schematic diagrams for all active bandpass filters.

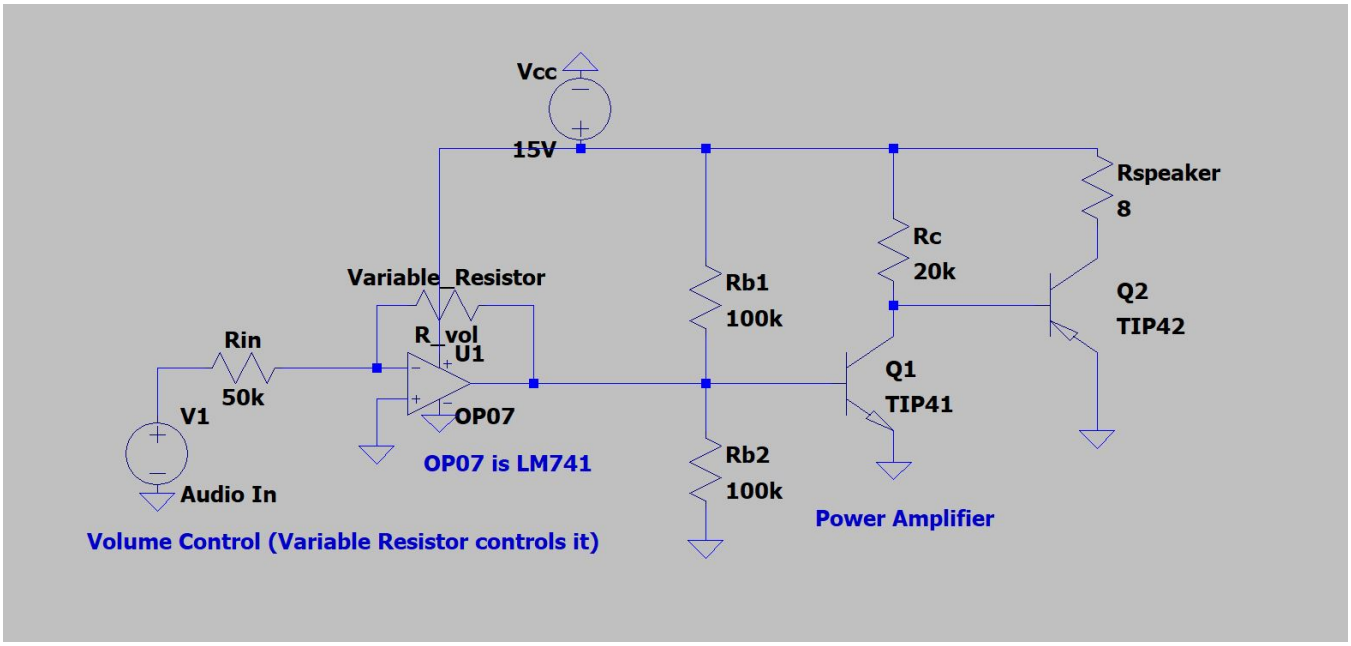


Figure 3: Left: a regular inverting amplifier for volume control. Right: a Sziklai pair using power transistors to drive the 8 Ω speaker.

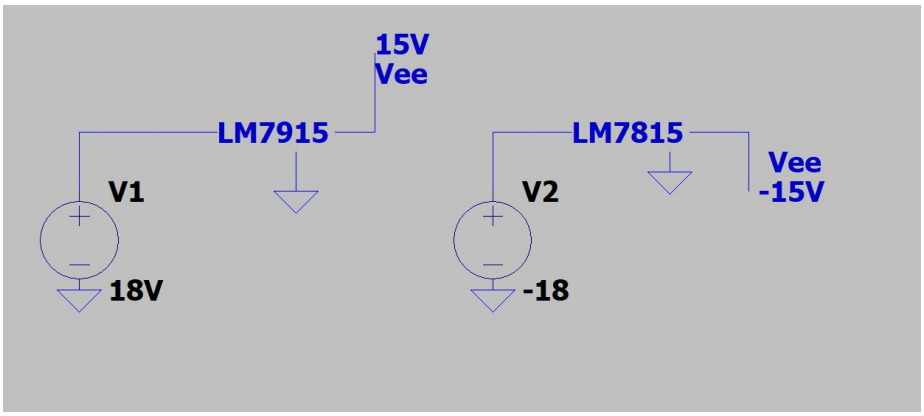


Figure 4: Voltage Regulator

5 Arduino script

For the purposes of this project, we were allowed to use an Arduino to do processing outside of the filtering (which had to be done in analog). We decided to use the Arduino to perform thresholding on the filtered signals to determine when to light the LEDs.

The Arduino script is very simple: for each filter output, choose a voltage threshold (empirically determined) above which the LED should turn on. The filter outputs are sampled uniformly in discrete time and used to determine whether the corresponding LED should turn on or off. Note that this simple sampling method doesn't detect extremas or zero-crossings, but is rather a random sampling of the possible voltages given the instantaneous signal amplitude. As a result, a hysteresis delay factor is used so that the LEDs don't immediately turn back off.

The Arduino is then connected to three digital outputs, which are connected to LEDs in series with a $220\,\Omega$ resistor. The schematics are not shown here because they are trivial.

An alternative (using analog components) would be to use a comparator (LM311) on the smoothed rectified output. We did not have enough time to try out this method, but we estimate that it will give better results than the Arduino's method due to the limited sampling rate of the Arduino, as compared with the high slew rate of the comparator and the continuous (smoothed) voltage.

```
int val = 0, iThreshold = 20, i = 0, j;

int inputs[] = {A0,A1,A2},
  outputs[] = {2,3,4},
  vThresholds[] = {25, 130, 60},
  lastIs[] = {0,0,0},
  ons[] = {0,0,0};

void setup() {
  for (j=0; j<3; ++j) {
    pinMode(inputs[j], INPUT);
    pinMode(outputs[j], OUTPUT);
  }
}

void loop() {
  ++i;

  for (j=0; j<3; ++j) {
    val = analogRead(inputs[j]);

    if (val > vThresholds[j]) {
      lastIs[j] = i;
      if (!ons[j]) {
        digitalWrite(outputs[j], 1);
        ons[j] = 1;
      }
    } else {
      if (ons[j] && i-lastIs[j]>iThreshold) {
        digitalWrite(outputs[j], 0);
        ons[j] = 0;
      }
    }
  }
}
```

Figure 5: Thresholding script for Arduino. `vThresholds` are found empirically and are (unfortunately) sensitive to environmental conditions. The counter variable `i` and the corresponding variable `lastIs` are used for hysteresis.

6 LTSpice simulations

See the simulations for each filter in Figures 6, 7, 8. An AC sweep is performed to simulate the frequency response of these filters.

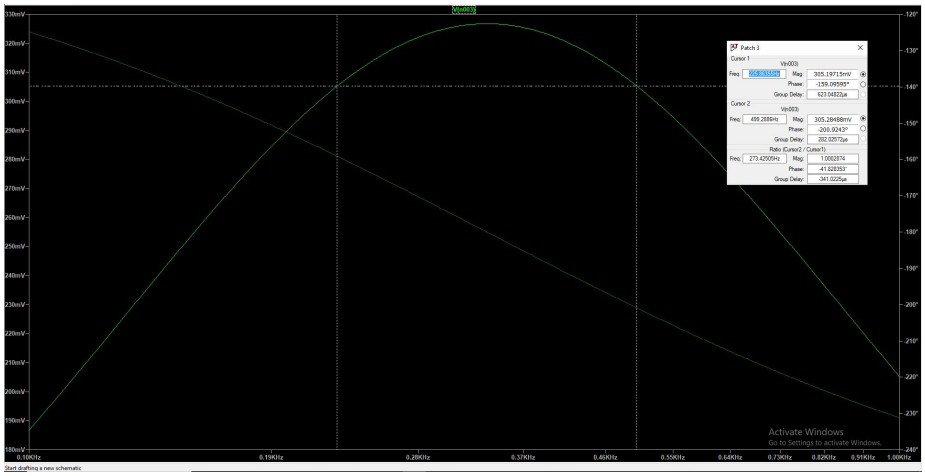


Figure 6: Simulation of Bass Filter

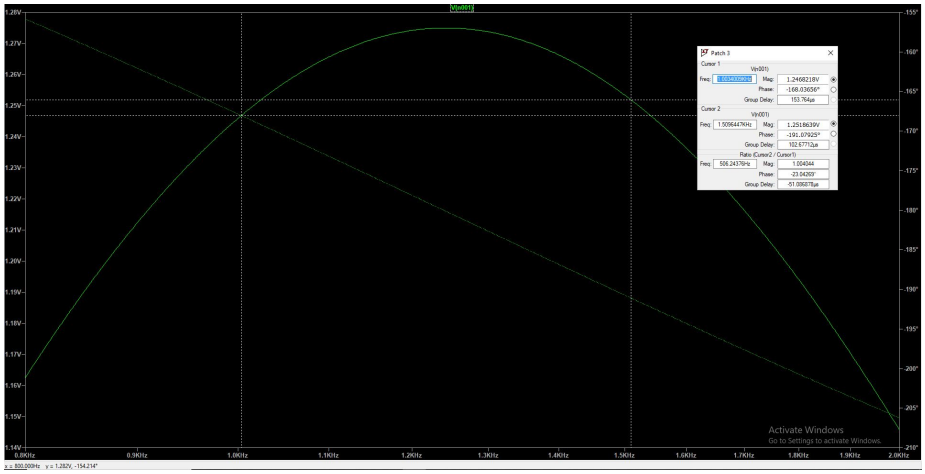


Figure 7: Simulation of Midrange Filter

In Figure 6, we see that cutoff frequency that we want (225 Hz and 450 HZ) is about (350 mV), which is where we would be setting our threshold to detect. There will be a little bit of error as about 500Hz gets included within

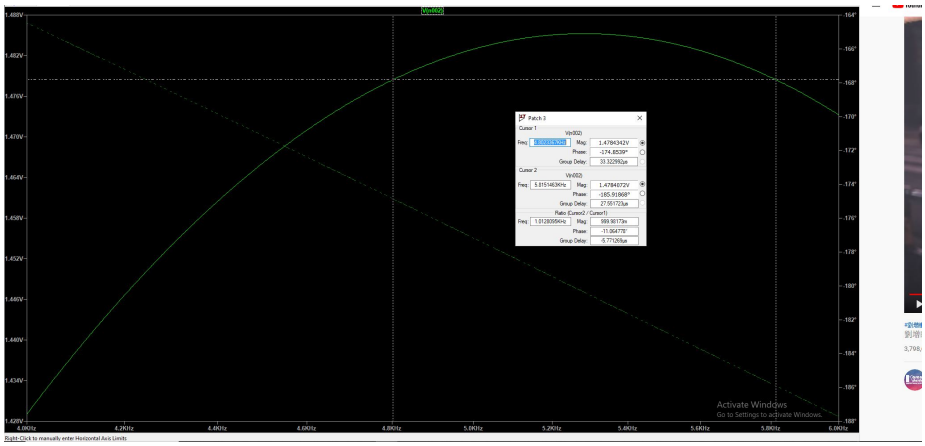


Figure 8: Simulation of Treble Filter

the range, but that should not affect the overall result as it will definitely not include frequency value that are not close to the thresholds.

In Figure 7, we see that the cutoff frequency that we want (1000Hz - 1500Hz) about 1.24-1.25V. Which is the threshold voltage we will be setting for our design. As stated with the Bass filter, there will be a little error around the threshold and probably, about $\pm 5\%$ around the threshold.

In Figure 8, we want the range to be around 4800Hz to 5500Hz. From the simulation, we see that the range is 4800Hz and 5800Hz. It is off due to we want to hold the voltage gain under 3, and thus we do now want it to be super high and make it hard to set threshold for our Arduino program.

7 Constructed Device

The end result is shown in Figure 9. Detail of the breadboard is shown in Figure 10. It is unfortunately messy due to the lack of time needed to refactor and tidy it up. Details are given in the figure captions.

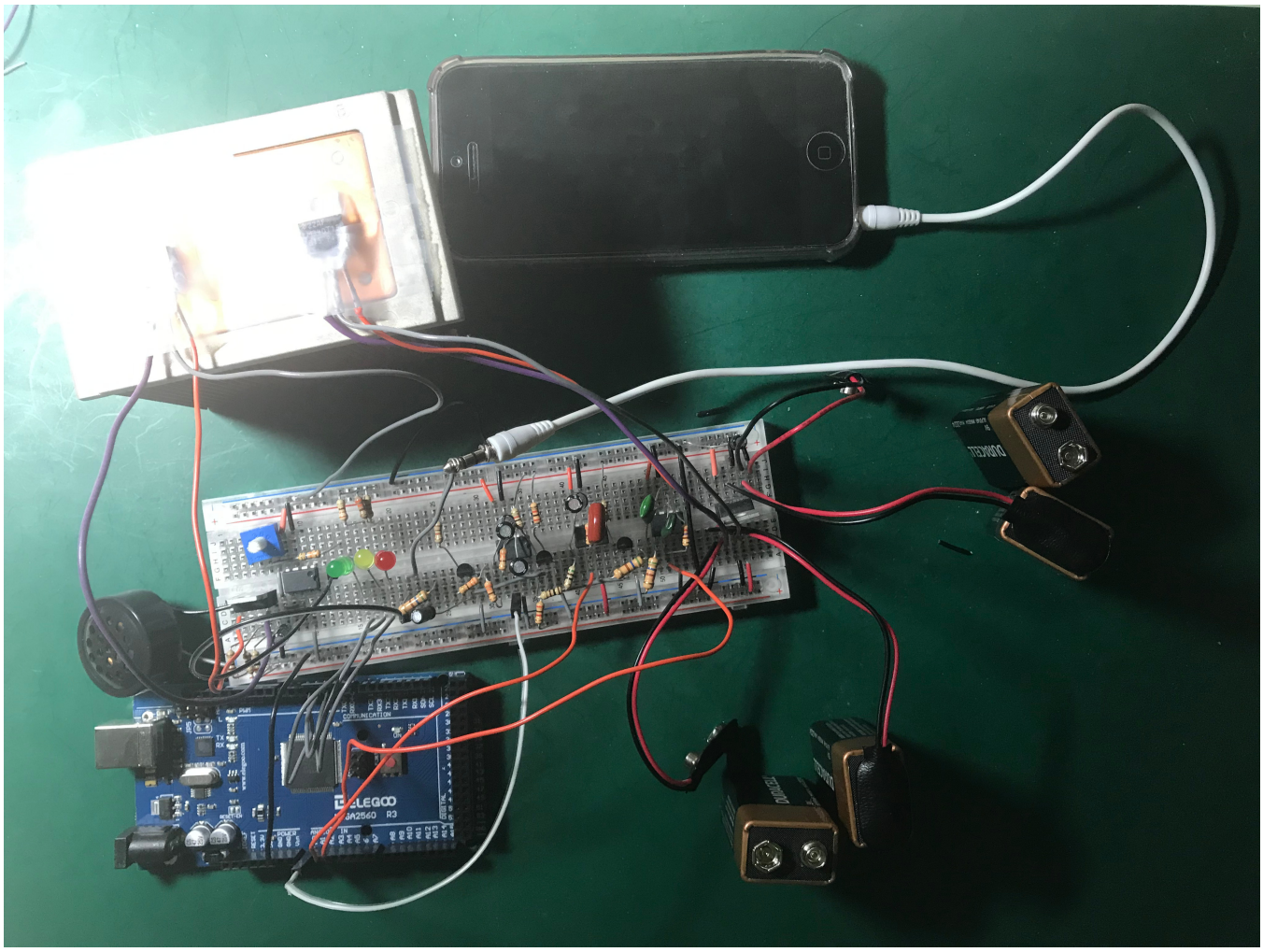


Figure 9: Construction. Notes on the non-breadboard components, clockwise from top left: The TIP42 and LM7815 are (Scotch-taped) attached to a CPU heatsink from an old PC. An iPhone 5 was used as the audio source. Four 9V batteries were used to power the power rails via voltage regulators. The Arduino takes as inputs the (analog) filter outputs and outputs (digital) LED signals. The speaker is an 8 Ω speaker of unknown wattage.

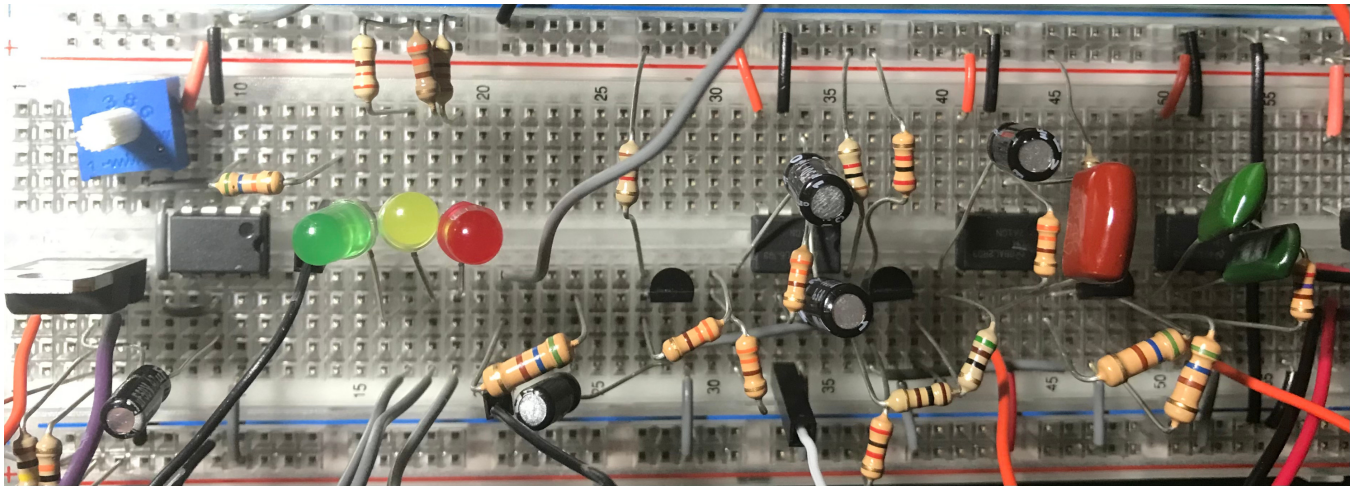


Figure 10: Detail of breadboard in construction. The top rails are GND and VEE; the bottom rails are GND and VCC. On the bottom left, the power amplifier (TIP41 is shown, TIP42 is on the copper heatsink). On the top left (potentiometer and LM741) is the volume control. The input to the volume control (top, row 14) is connected via black jumper cable to the audio input (bottom, row 21), which is also the input to the filters. The three LEDs are used for the color organ output and are triggered by the Arduino via the gray wires on the bottom. The parts to the right of row 55 are the power regulator circuits. The remaining circuits are the three filters: each contains a buffer stage (2N3904 emitter-follower) followed by an active bandpass filter (LM741 op-amp with two capacitors and two resistors – see the schematic diagrams for exact values). There is only one biasing resistor pair on row 36 (two 2 kΩ resistors between VCC and VEE; all three buffers share this same buffer, by connecting their bases together, current-mirror style. The outputs of the filters are jumped to the Arduino analog pins.

8 Evaluation of Constructed Device

The two lower-frequency filters perform decently. This means that, assuming no changes in environmental factors (e.g., temperature) and music amplitude, the filters and the thresholding cause the LEDs to light when the audio is in the correct frequency range. We tested on a YouTube video[5] that performed a frequency sweep at constant amplitude under a short period of time, which fulfills most of these assumptions. This is sufficient for our academic curiosity; however, it would not be very good in a practical setting. A professional color organ should be more robust to noise, changes in music amplitude, changes in environmental factors, and multiple tones.

However, we were unable to get the highest (treble) filter to light the LED at any threshold. We ran low on time and were unable to troubleshoot this. Given that the other two work, we assume this is some fault of our building rather than some fundamental flaw.

There is not much to evaluate about the speaker: the sound is good but it is very power inefficient. It would be a good idea to study the common topologies of the different power amplifier classes before a second attempt at building a power amplifier.

9 Conclusions

The implementation of the project was a lot more difficult than what we expected, as we see that there are problems regarding our third filter on implementation (no detection).

Another key problem about our project is that it should work if the input signal has varying amplitude (mixed volume) as our threshold system using the Arduino is very sensitive to input amplitude. For future work, we would try to make the system robust to any amplitude of audio (perhaps by normalization) and other potential abnormalities in the noise.

Lastly, we did not implement threshold using analog logic, instead doing it with an Arduino. We believe that using a comparator would be better, so that is another thing that we should look into for further implementation.

[5] `https://www.youtube.com/watch?v=qNf9nzvnd1k`