I've always had a fascination with flashing lights; especially when the lights are synchronized to music. Long time readers of Nuts and Volts probably realized this about me a long time ago. Because of this fascination I have built more varieties of color organs than I should probably admit. If you are unfamiliar with the term, a *color organ* also called a *light organ* is defined by *Wikipedia* as:

> A circuit that separates the audio signal into frequency bands and controls the light channels according to the average level of each band. A typical party light organ of the 1970s had three spotlights, red, green and blue, for sounds in bass, medium frequency and high frequency.

Due to their simple design, analog color organs have been popular as DIY projects for electronics hobbyists for years and can still be found for sale as kits on the Internet today. So this proves I'm not the only one fascinated with flashing lights. Unfortunately most analog color organs don't work very well and all of the colors end up flashing nearly all of the time regardless of the music being listened to. What I was after when I designed this color organ was to have good frequency selectivity so that the color and brightness of the display LEDs are directly related to the frequency content of the music; not just to its amplitude.

If one were to pull out the January 2008 issues of Nuts and Volts you would find an article I wrote about the design of the first *digital* color organ I ever developed which used digital signal processing (DSP) technology in place of traditional analog filtering. I still get emails from people asking me if the design I presented in those articles have been updated because they are still interested in building it. Well I have good news for those folks. In this article I will present an all new design for a color organ with features unavailable in any other color organ I am aware of. Briefly this new design:

1. Uses a string of 120 WS2812B RGB LEDs (also called NeoPixels) for the lighting element
2. Has a touchscreen LCD display which is used to configure the color organ's operation
3. Has a built-in automatic volume control which allows the color organ to respond correctly given changes in volume of the input musical material.
4. Uses a 1024 point Fast Fourier Transform or FFT to divide the audio spectrum into eight musically related octave bands
5. Allows user specified control of the density of LEDs used per band.
6. Special dynamics mode periodically reassigns the LEDs for each band which, in effect, changes the physical layout of the LEDs.
7. Uses the Teensy 3.1 micro controller (uC) and software written in the Arduino environment.

In a nutshell, this color organ digitizes audio from either a microphone or from line level inputs, sends it through the automatic volume control (if enabled) and then feeds it to a 1024 point FFT. The FFT (described a little later) breaks down the complex digitized audio into frequency bins and subsequent processing transforms the bins into eight frequency bands which are used to drive a string of WS2812B RGB LEDs.

Hardware

The hardware for the color organ consists of the major parts described below.

Device	Function
Teensy Audio Adapter	This part handles the digitization of audio coming from either a microphone interface or stereo line inputs. It has a small number of analog components necessary to bias the off board condenser microphone. It also provides a micro SD card interface supporting a FAT type filesystem.
Touchscreen LCD display	All user interaction with the color organ (with the exception of turning the color organ's power off and on) is done via a touchscreen LCD display.
Teensy 3.1 micro controller (uC)	All of the color organ's functionality is provided by software running in this ARM based micro controller. The 32 bit Teensy 3.1 has 262144 bytes of flash memory for program/data storage and 65536 bytes of RAM.
WS2812B (NeoPixel) RGB LED string/strip	The display aspect of the color organ is provided by a string of 120 RGB LEDs. Each LED is individually addressable and each provides its own Pulse Width Modulation (PWM) controller for varying color and brightness. These LEDs are capable of a wide range of colors and extreme brightness.

Figure One shows the schematic diagram of the color organ and the following is the parts list.

Designation	Description	Source
Teensy Audio Adapter	Audio Adapter	pjrc.com
Micro SD card	Micro SD memory card 1 MB or greater	eBay, Amazon
Teensy 3.1 uC	Teensy 3.1 micro controller	pjrc.com
PJRC Color Touchscreen LCD	320x240 LCD display and touchscreen	pjrc.com
Line in Left, Line in Right	RCA audio jacks	Radioshack
Electret Condenser Microphone	Miniature microphone element	pjrc.com
Power cord	Standard 110 volt cord	Home Depot
Power switch	SPST off/on switch	anywhere
Power supply	5 VDC @ 3 amps minimum	eBay

Designation	Description	Source
74AHCT125	Quad level converter	adafruit.com
Program pushbutton switch	Miniature momentary pushbutton switch	eBay
WS2812B (NeoPixel) LED strip	5 volt with 30 LEDs per meter density. Need 120 LEDs total.	eBay
C1	470 uF 16 VDC or greater electrolytic capacitor	Radioshack
C2, C4	.1 uF ceramic bypass capacitor	Radioshack
C3	1000 uF 35 VDC or greater electrolytic capacitor	Radioshack
R1, R2	100 ohm ¼ watt resistor	Radioshack
Miscellaneous Electronic Hardware	Wire, shielded cable for inputs, solder, screws, standoffs, etc.	anywhere
MDF for the cabinet	1/2” medium density fiberboard	Home Depot
Prismatic Plastic Sheeting	Prismatic Plastic Sheeting	Home Depot
Miscellaneous Cabinet Hardware	Screws, glue, rubber feet, etc.	Home Depot

I built my color organ using point to point wiring. Figures Two and Three show the front and back of the circuit board. The Teensy 3.1 is connected to the Audio Adapter with header pins that extend from the adapter through the Teensy's PCB then through the pref board. This makes all of the necessary connections between the uC and the audio adapter. The micro SD card should be inserted into the connector on the Audio Adapter once the adapter and the Teensy 3.1 are soldered together and wired onto the circuit board.

Software

The code which implements the color organ is unfortunately rather complex and is made up of many files. Fortunately you don't really have to understand it in detail to use it. A much deeper understanding is necessary, however, if you plan to modify the code.

The color organ program is made up of the following files:

File Name	Function
BaseScreen.h	This is the base class of all of the individual screens used in the LCD user interface. It defines functions that each screen must implement and provides some low level processing of touchscreen events.
Colors.h	Code in this file is used to generate 16 bit color values for use on the LCD display and 24 bit color values for sending to the RGB LED string.

File Name	Function
GUI_Elements.h	Code in this file defines all onscreen widgets that users interact with. This includes text fields, VU fields, buttons and value pickers. In addition it has the *drawBorder* function which provides the overall look of each of the user interface screens.
LED_OutputControl.h	This code provides the software interface to the string of RGB LEDs. Functions in this file assign LEDs to each of the eight bands according to the currently configured LED density and control the colors used for each band. Band colors are assigned across the visible spectrum in ROY G BIV order (red, orange, yellow …) with red at the bass end and violet at the high end. Every time the *assignLEDsBasedOnDensity* function is called the LEDs assigned to each band are randomly reassigned changing their physical location along the LED string.
PrintExtensions.h	Code for formatting text strings within bounded rectangular areas of the LCD display.
Screen_Config.h	Code in this file implements the *Configuration Screen* shown in Figure Five. Touching buttons on this screen cause various other screens to be displayed. Touching the *Back* button causes control to return to the Main Screen.
Screen_Density.h	It is on the *Density Screen* (Figure Seven) where the density of LEDs used for each band is configured. Each of the eight bands have a maximum of 15 LEDs. If a density of 50% is selected, half of the LEDs in each band will be used for display. If 25% is selected, a quarter of the LEDs in each band will be active.
Screen_Files.h	Code in this file allows the current configuration of the color organ to be stored and loaded from the micro SD card plugged onto the Audio Adapter. See Figure Eight. To save a configuration, first select the file to use and then touch the *Store* button. To load a configuration, first select the file and then touch the *Load* button. Configuration files stored on the SD card are never deleted but are meant to be overwritten as necessary.
Screen_Main.h	The *Main Screen* (Figure Four) is what is displayed when the color organ is first powered up. This screen has two VU like meters that display the audio levels being seen by the color organ. If this screen is touched the *Configuration Screen* is brought up.
Screen_Misc.h	The *Miscellaneous Screen* (Figure Six) allows selection of the audio input; either Mic or Line, controls whether the Automatic Volume Control is engaged or not and allows setting the time interval in minutes between dynamic changes in LED band assignments. If the interval is set to zero, the LED assignments are static and never change.
TeensyColorOrgan-FFT.ino	This is the main Arduino program that coordinates all actions of the color organ. It defines all objects used in code and a Finite State Machine (FSM) which controls the overall operation. This code is structured so that the color

File Name	Function
	organ remains operational while the user is interacting with the LCD touchscreen display and so that changes made by the user are reflected in the color organs' operation immediately.
Readme.txt	This file has instructions for compiling the code in the Arduino environment and for programming the Teensy uC. Article length restrictions prevent me from including these instructions in the text.

The core of the color organ code is the FFT (the FFT code is provided by the Audio Adapter library) that converts the digitized audio in the time domain into the frequency or spectral domain. That is, the audio is broken out into its component frequencies. Since the audio is digitized at a 44,100 samples per second rate and there are 1024 points in the analysis each bin is 44,100 / 1024 or 43.066 Hz wide with there being 513 usable bins. There are an additional 512 bins that are deemed “unusable” because they represent frequency above the Nyquist frequency (see https://en.wikipedia.org/wiki/Nyquist_frequency for the details) which is half the sample rate. The output of most FFT algorithms are bins of complex numbers containing real and imaginary parts (for an explanation of complex numbers see https://en.wikipedia.org/wiki/Complex_number). The pjrc.com Audio Adapter library converts these complex numbers and returns a single floating point number representing the magnitude of the frequency content of each bin.

In a FFT spectrum analyzer application the magnitude of each bin would be displayed on a screen which would give a picture of the frequency content of the signal being analyzed. In our application we need to further process the FFT data to make the color organ react in a more musical way. We do this by lumping bins together into eight frequency bands that are related by octaves, a very musical quantity. The table below shows how the FFT bins are converted into the eight frequency bands.

Band Number	Starting Bin Number	Ending Bin Number	Number of Bins	Minimum Band Frequency	Maximum Band Frequency
0	1	2	2	43	86
1	2	4	3	86	172
2	4	8	5	172	344
3	8	16	9	344	689
4	16	32	17	689	1378
5	32	64	33	1378	2756
6	64	128	65	2756	5512
7	128	256	129	5512	11025

As you can see frequencies below about 43 Hz and above about 11025 Hz are ignored in the current design.

So whenever new FFT data is available within the color organ program, the conversion from bins to bands is performed and the resultant values are used to drive the corresponding band LEDs on the LED string.

Packaging

Color organs have typically been built into some sort of cabinet or box making for an easily transportable package. That's what I decided to do but you might decide to do things differently. I could see attaching the 120 LED string to the underside of your stereo table or mounting it around your speakers. The LED strip could even be mounted in a coffee table. You also have the choice of viewing the LEDs directly or diffusing their light by placing them behind prismatic sheeting as I did. The prismatic sheeting causes each LED to project a halo like pattern which I think enhances the display.

I decided to use MDF (medium density fiberboard) for my cabinet as I had it available in my shop. I also decided that the cabinet would initially be painted but may eventually be veneered if necessary for placement in the living room. To determine the size of the cabinet I first had to decide on the physical arrangement of the LED strips to be used for display. Since I had bought LED strips with a 30 LEDs per meter density I decided to arrange the LEDs into eight strips of 15 LED each. Adding a uniform border distance around the LEDs led to the size of the back panel on which the LEDs were to be mounted (they have an adhesive backing) and eventually to a cabinet size of 16 1/8” x 23 1/2” x 6 1/2”. The dimensions of the cabinet parts are shown in the table below:

Description	Size	Material
Cabinet Top	16 1/8” x 6 1/2” x 1/2”	MDF
Cabinet Bottom	16 1/8” x 6 1/2” x 1/2”	MDF
Cabinet Left Side	22 1/2” x 6 1/2” x 1/2”	MDF
Cabinet Right Side	22 1/2” x 6 1/2” x 1/2”	MDF
Cabinet Back	16 5/8” x 22 1/2” x 1/8”	Masonite
Cabinet Screen	16 5/8” x 22 1/2” x 3/32”	Prismatic Plastic Sheet

The individual machined pieces can be seen in Figure Nine. The top and bottom of the cabinet are screwed onto the sides to hold things together. You can see the screw holes in the top and bottom pieces. I used my table saw blade to cut 1/4” deep slots in each piece so that the back of the cabinet and the prismatic plastic screen are embedded into the cabinet on all sides making for very sturdy construction. The slot for the back is slightly wider than the slot for the screen due to the difference in material thickness. You can also see the cutout I made for the touchscreen LCD and the holes used to mount the color organ's circuitry.

Figure Ten shows the pieces of the cabinet put together to test their fit. Figure Eleven show the cabinet after I painted the inside a dull black and the outside an off white color. Figure Twelve shows the beginning of the assembly process as I have mounted the power cord, power switch, power supply, the

line input jacks and the eight 15 LED lengths of the LED string. Figure Thirteen is a top view of the color organ's internals while they are working. As can be seen, the LED strips are wired together using a green wire for the data connection between the strip sections and the silver bus wires are used to connect five volt power and ground to each section. Finally Figure Fourteen shows a rear view of the working color organ.

Conclusions

If you like color organs as much as I do and you have been looking for one to build, this project might just be for you. If you can read a schematic and solder you shouldn't have any trouble building the hardware and since the software is available on the Nuts and Volts website you should be good to go.

The eight channels of frequency selectivity makes for a very lively, music oriented display with all types of music. Having a built in microphone means that you can take the color organ with you and have it running in minutes without having to connect anything except power.

Here are some other ideas on how you might use the hardware and software provided with this article:

1. Wrap your Christmas tree with the WS2812B LED string and connect it to the color organ circuitry. That way your tree can pulse with your favorite Christmas music.
2. Build the color organ circuitry and the WS2812B LED string into a coffee or end table.
3. The color organ code could be changed slightly to function as an audio spectrum analyzer
4. The color organ code could be changed for fewer or more than eight bands of frequency response
5. The number of LEDs driven can easily be changed to add more LEDs or to reduce their numbers.

Resources

The code for this project should be available on the Nuts and Volts website after this article is published. All of the libraries required by the color organ are contained within the Nuts and Volts download. See the *Readme.txt* file for installation instructions. Sources for the required software components are provided below:

Software	Description	Source
Arduino Integrated Development Environment (IDE) Version 1.6.9	Code used to develop and/or change the color organ software and to program the Teensy 3.1 uC	https://www.arduino.cc/en/Main/Software
Teensyduino Version 1.29	Code which enables the Teensy 3.1 code to be developed within the Arduino IDE	https://www.pjrc.com/teensy/td_download.html
ILI9341_t3 LCD driver	Driver for the LCD display	https://github.com/PaulStoffregen/ILI9341_t3
XPT2046 touch screen driver	Driver for the touchscreen portion of the LCD display	https://github.com/PaulStoffregen/XPT2046_Touchscreen

Software	Description	Source
Teensy Audio library	Library which provides audio digitization and processing of audio	https://github.com/PaulStoffregen/Audio
Adafruit Neopixel library	Library for support of NeoPixel (WS2812B) LED strings	https://github.com/adafruit/Adafruit_NeoPixel
Teensy Color Organ code	The Arduino code which makes up the color organ	Nuts and Volts website at: http://www.nutsvolts.com or my github site at: https://github.com/CraigLindley/TeensyColorOrgan-FFT

Although not required for understanding the color organ presented here, the *Teensy Library Audio System Design Tool* is available online at: https://www.pjrc.com/teensy/gui/. This tool is used to design much more complex audio processing systems that use the pjrc Audio Adapter.

Finally, there are videos of the color organ in operation available on youtube.com at:

https://www.youtube.com/watch?v=fYzYyl-jh2g
and
https://www.youtube.com/watch?v=WyaueJN8Djo

Figure One

Teensy Color Organ Schematic

Craig A. Lindley - 2016

NOTES:

1. Connections between Audio Adapter and Teensy 3.1 are made when adapter is soldered onto top of the Teensy.
2. Only one section of 74AHCT125 used
3. Power cord, power switch, power supply and RCA line input jacks mounted to case and wired onto circuit board
4. Program pushbutton added because Audio Adapter hides button on Teensy

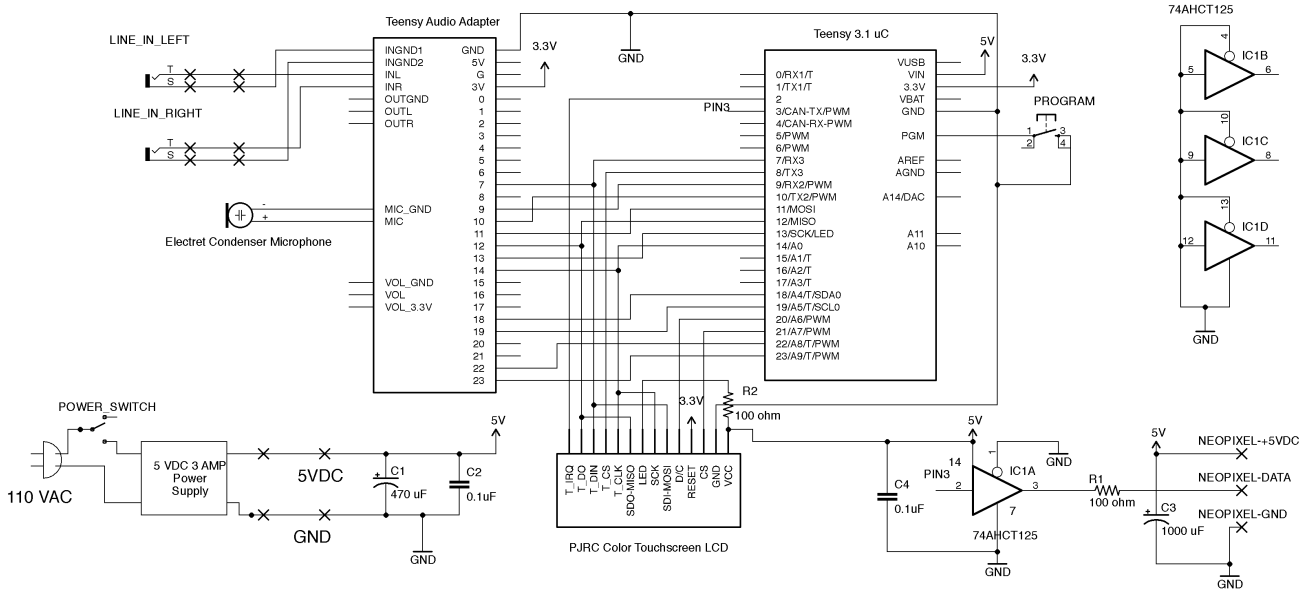

Figure Two
Teensy Color Organ Circuit Board – Front Side
The Teensy 3.1 uC is soldered beneath the Audio Adapter

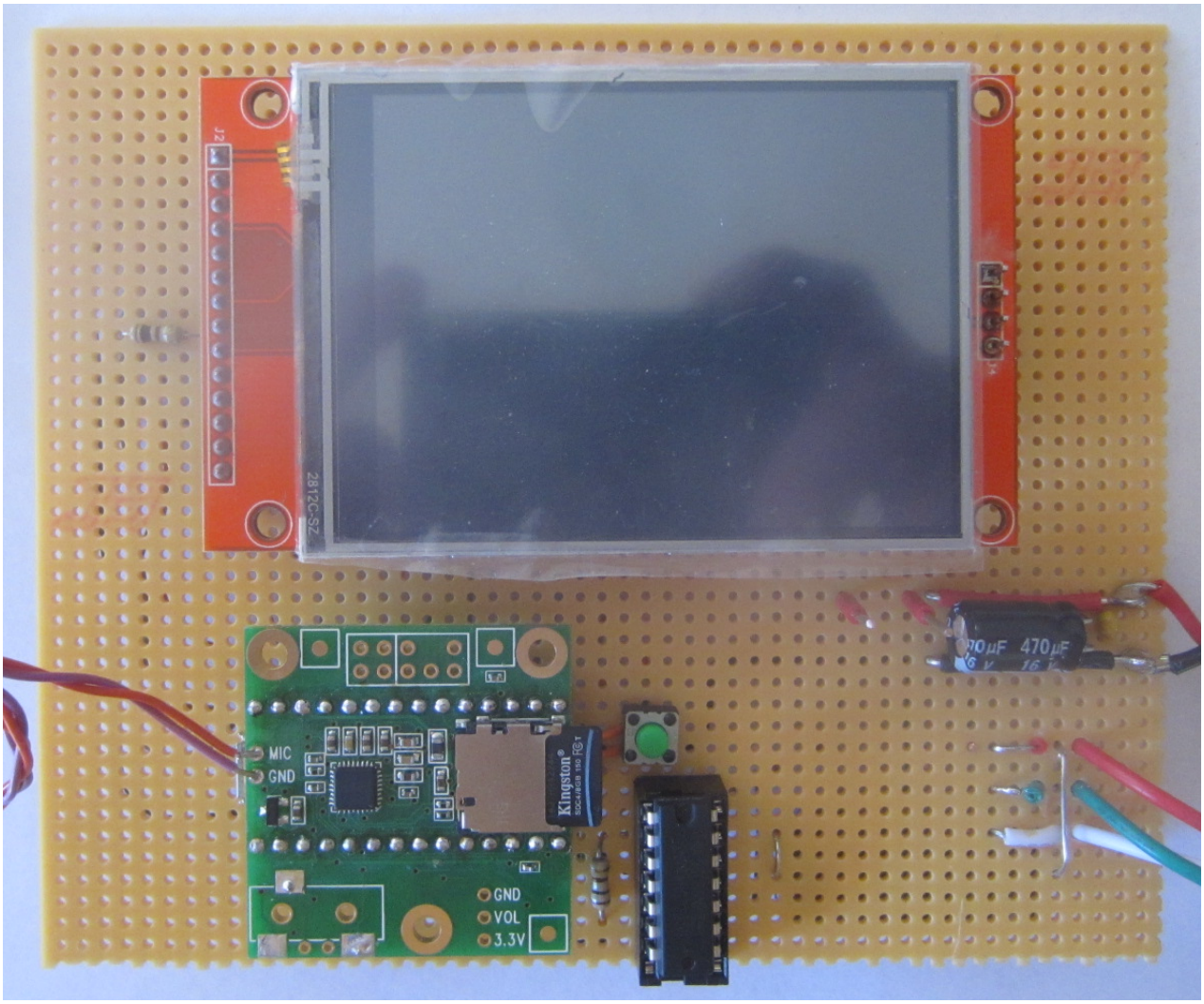

Figure Three
Teensy Color Organ Circuit Board – Back Side
Point to point wiring isn't pretty but is effective for one off projects

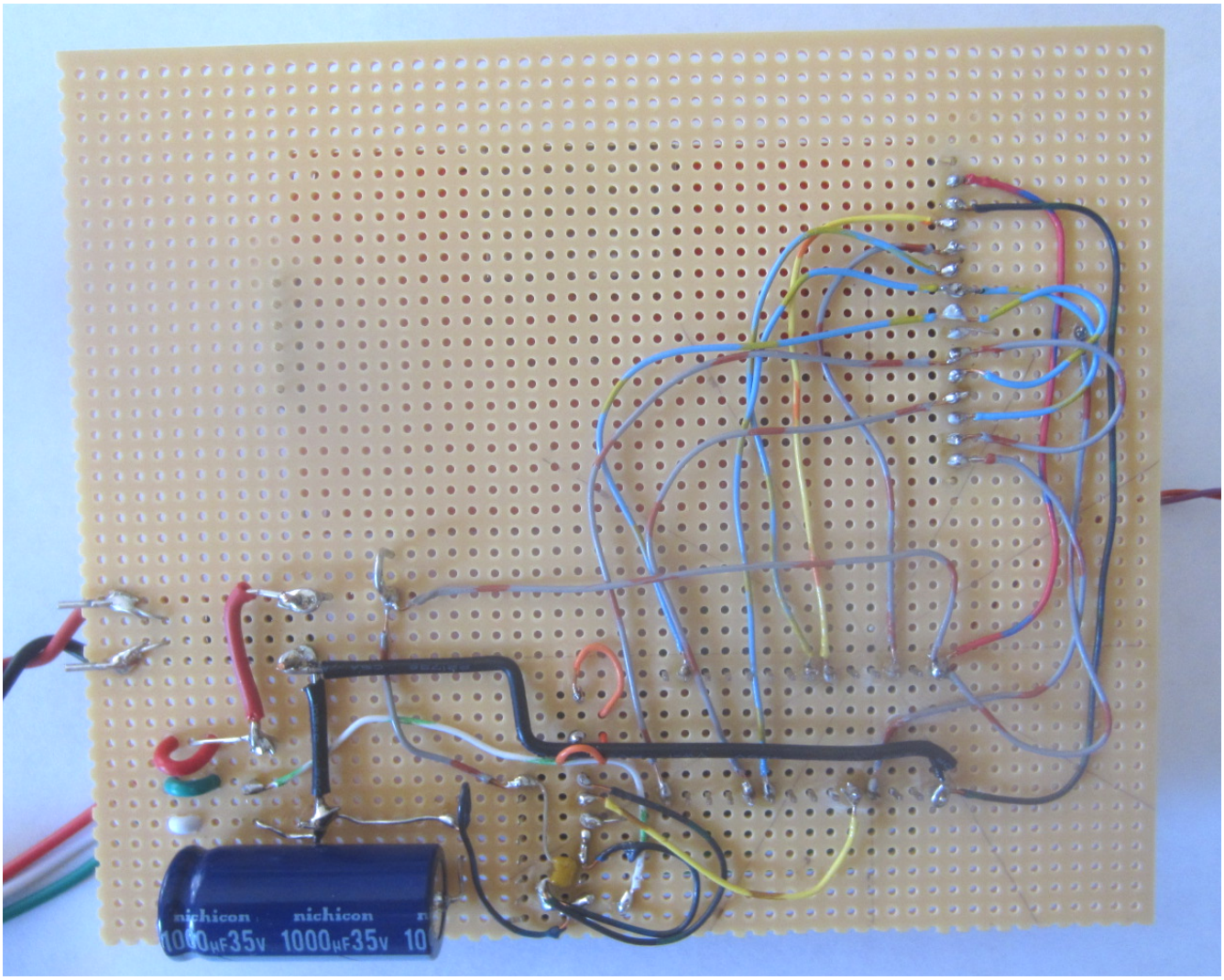

Figure Four
Main Screen showing Left and Right Channel VU Metering

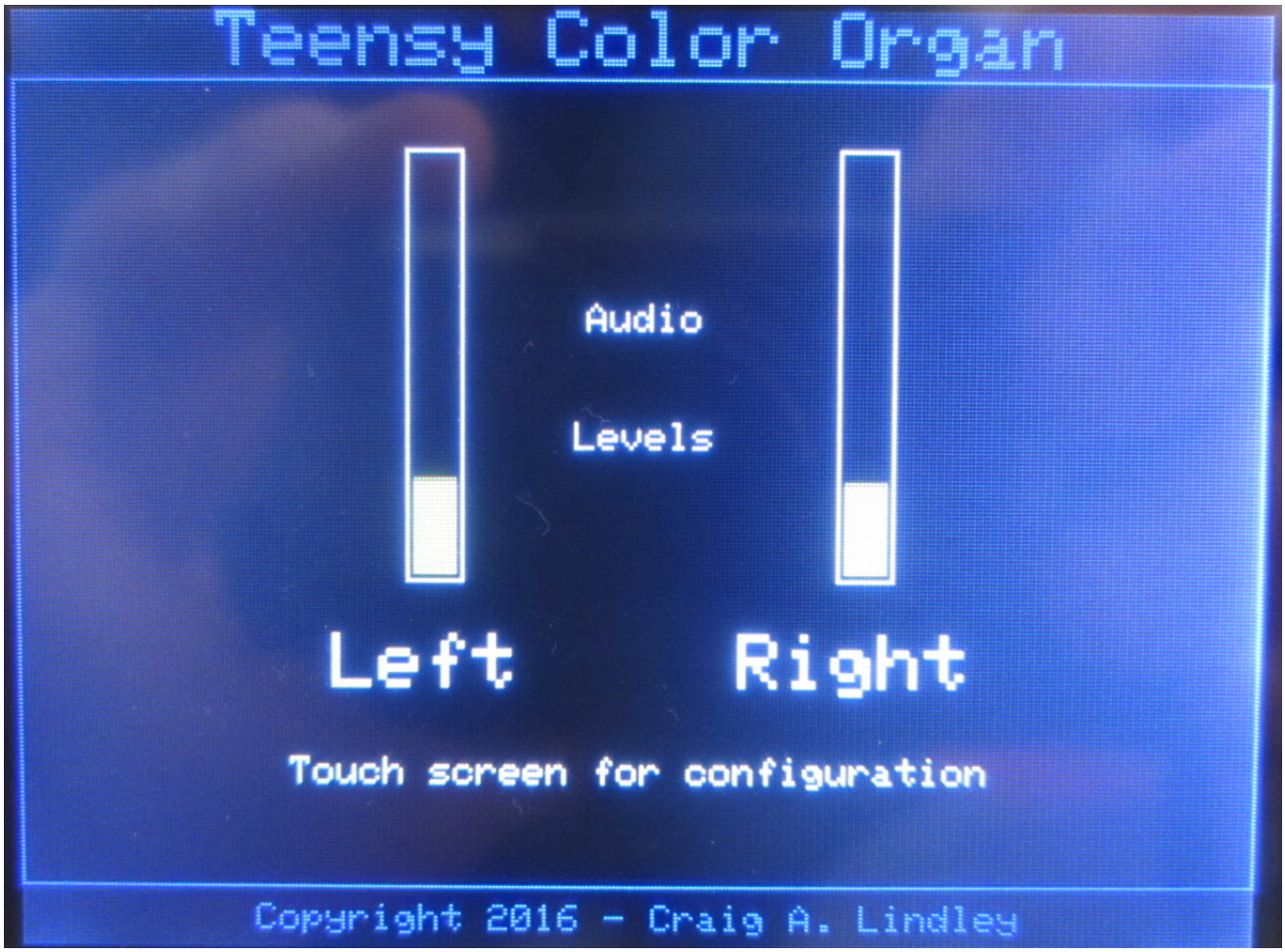

Figure Five
Configuration Screen

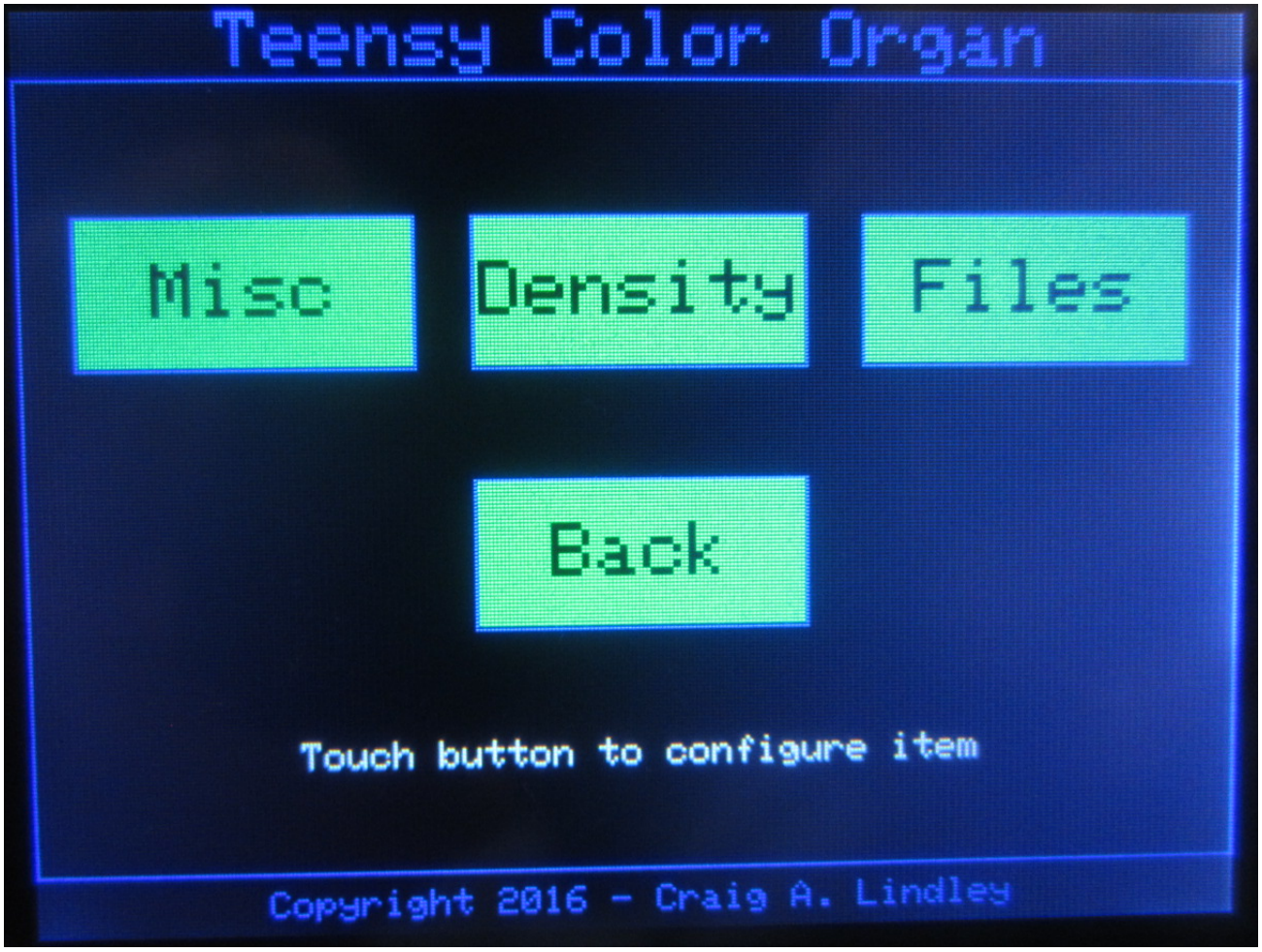

Figure Six
Miscellaneous Screen

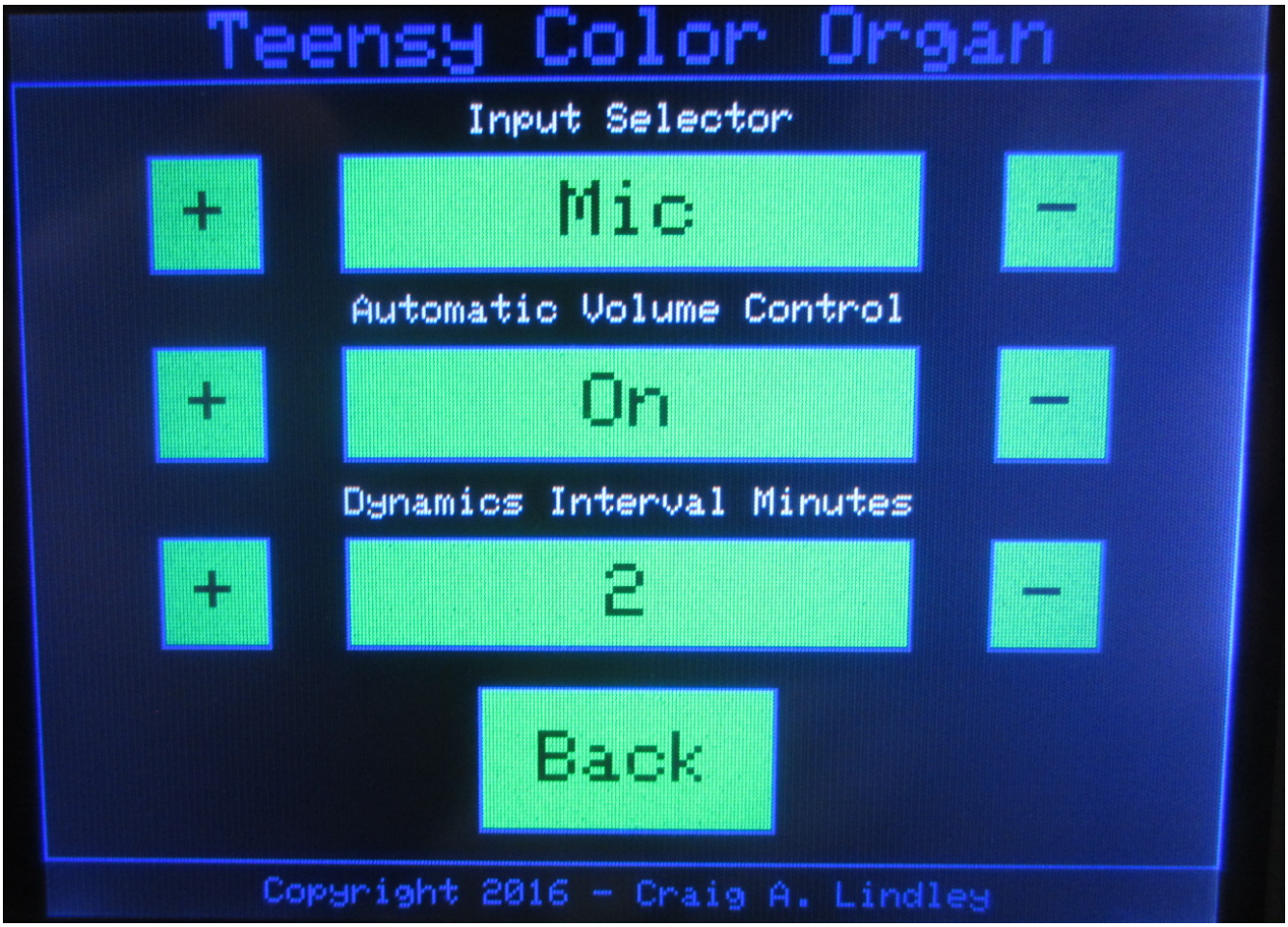

Figure Seven
Density Screen

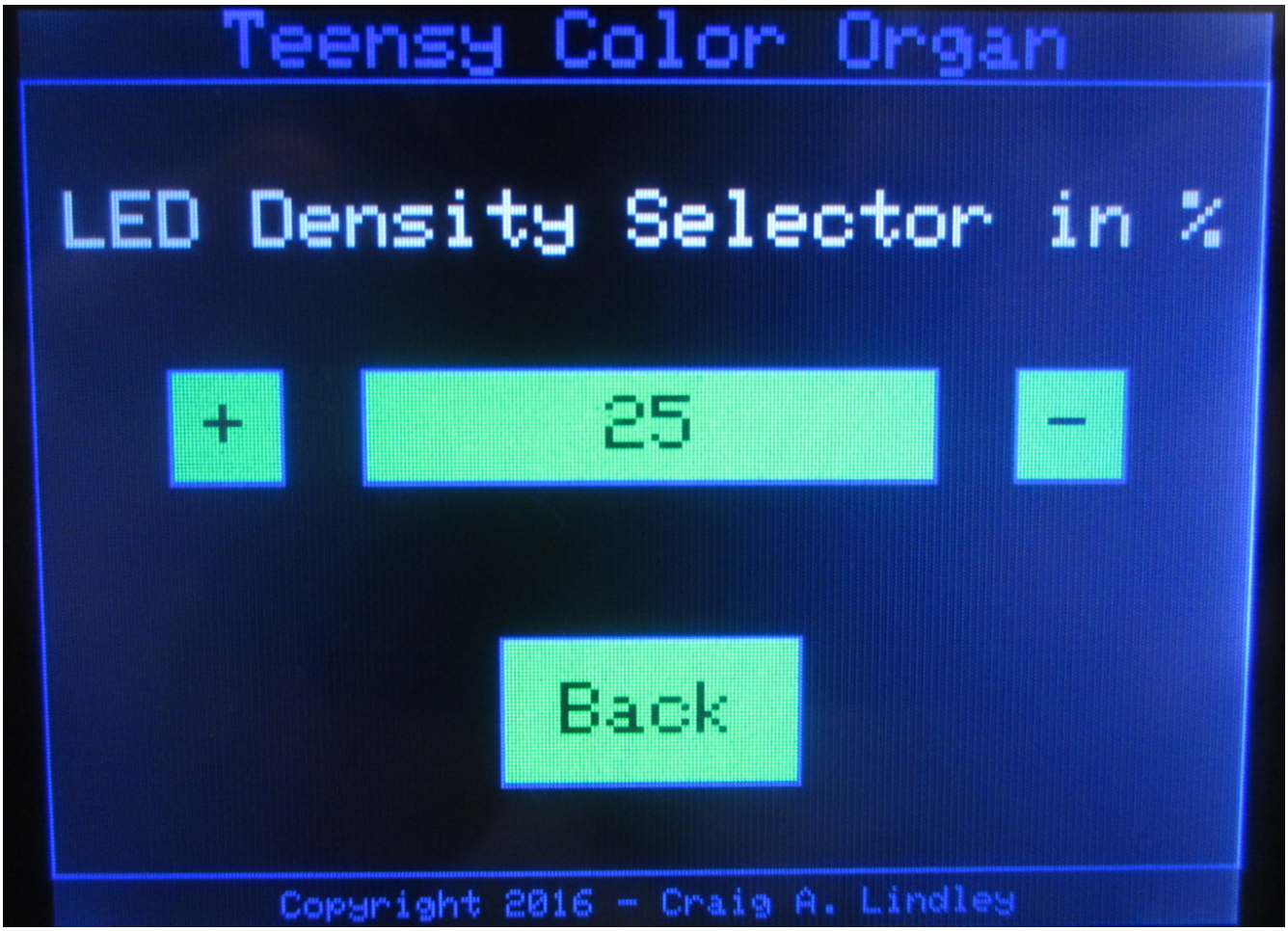

Figure Eight
Files Screen

Figure Nine
Machined MDF pieces for color organ cabinet

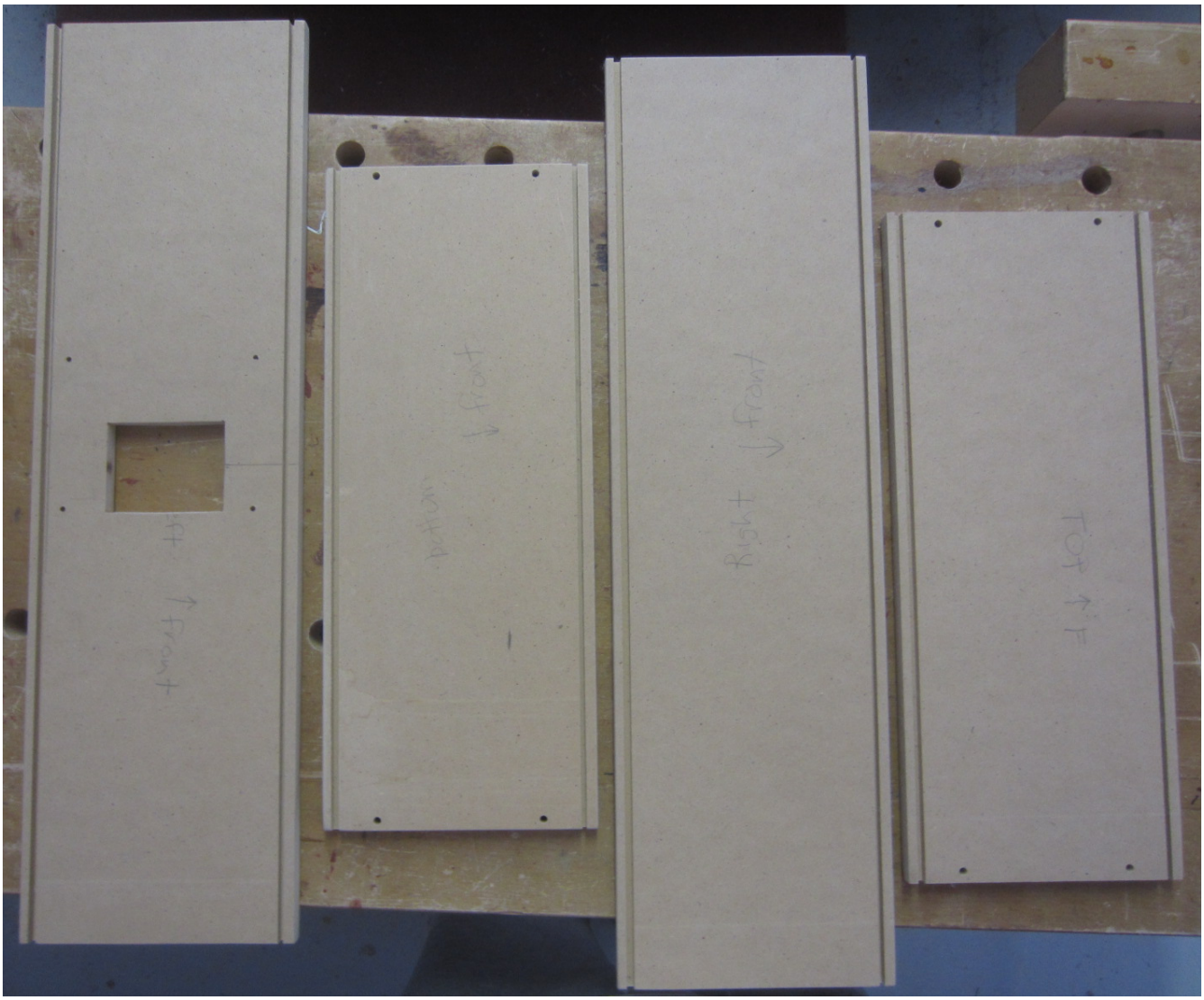

Figure Ten
Test fit of machined MDF cabinet pieces

Figure Eleven
Painted Cabinet
I painted the cabinet but I may veneer it in the future

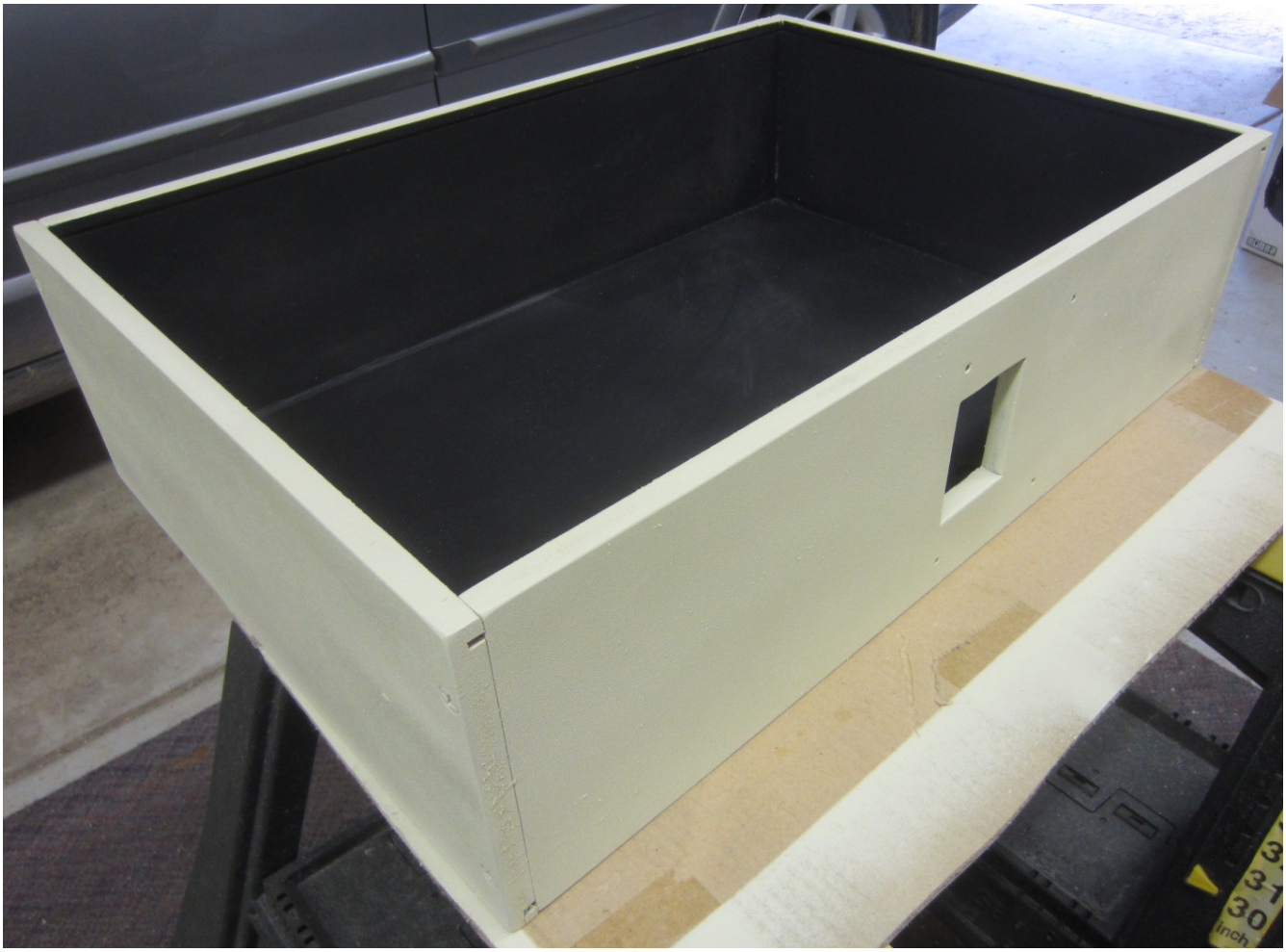

Figure Twelve
Beginning Assembly
LED strips, power supply, power switch and line input jacks in place

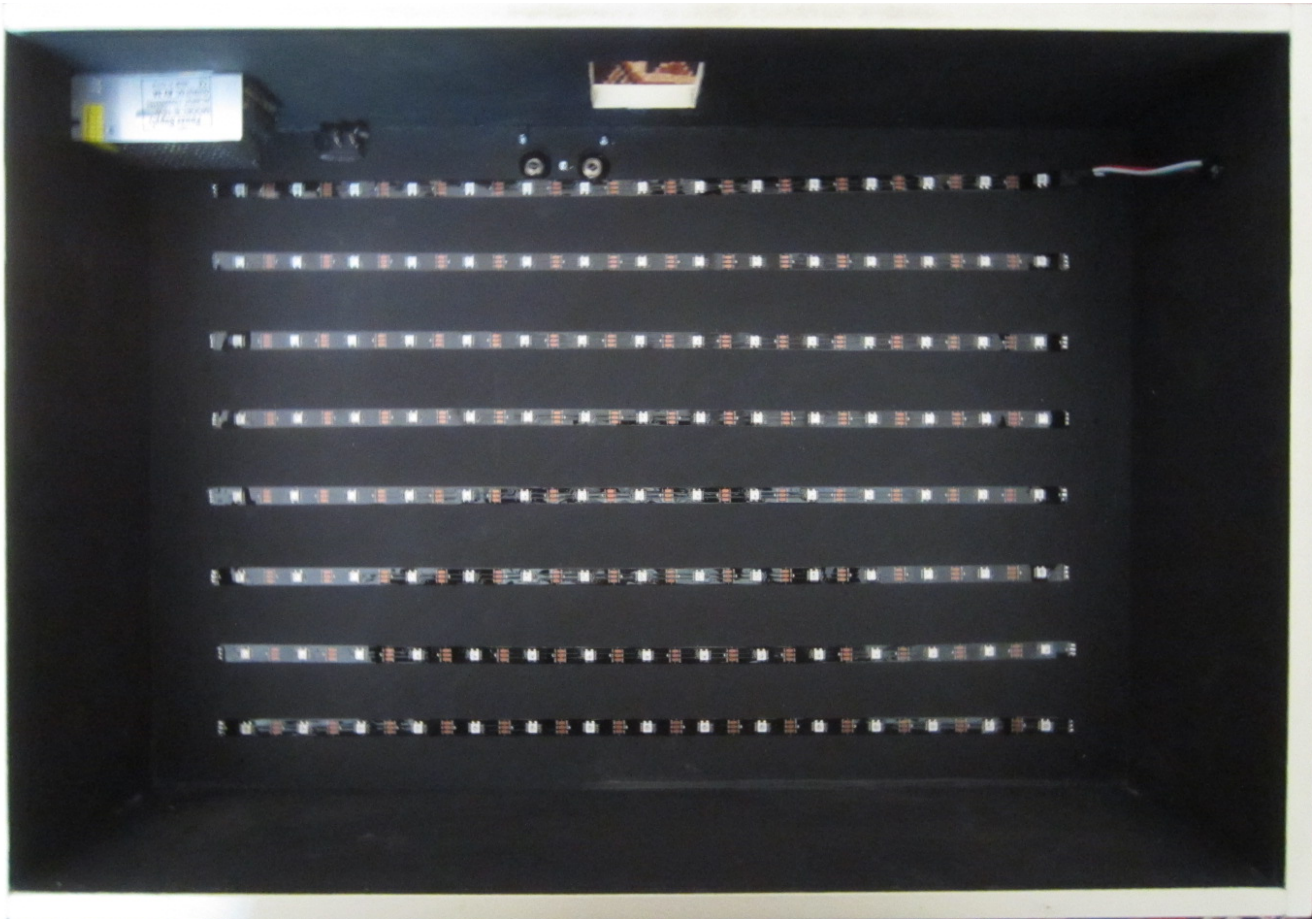

Figure Thirteen
LEDs in Action
Green wires connect the data inputs between strips and the silver wires connect the power

Figure Fourteen
Rear View of Cabinet
LCD, line inputs, power switch and power cord are visible