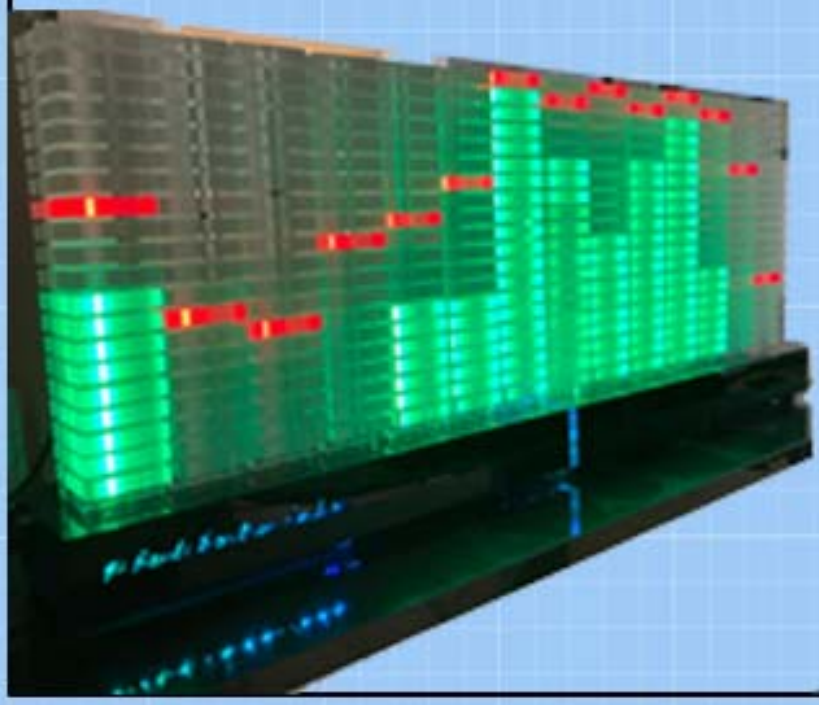
14 Ch Spectrum Analyzer

ARDUINO at its best



14 Channel
280 LEDS
40Hz - 16Khz
Line in
Microphone in
Different modes

Building Manual

M. Donners 11-04-2021

Table of Contents

M Donners Document version 3.0 d.d. 14-April-2021

1. Disclaimer and safety

I, Mark Donners, The Electronics Engineer, may or may not endorse various Do-It-Yourself (DIY) projects and all DIY projects are purely “at your own risk”. As with any DIY project, unfamiliarity with the tools and process can be dangerous. Posts should be construed as theoretical advice only.

If you are at all uncomfortable or inexperienced working on these projects (especially but not limited to electronics and mechanical), please reconsider doing the job yourself. It is very possible (but not likely) on any DIY project to damage belongings or void your property insurance, create a hazardous condition, or harm or even kill yourself or others.

I will not be held responsible for any injury due to the misuse or misunderstanding of any DIY project.

By using the information provided by me, (Website, YouTube, Facebook and other social media), you agree to indemnify me, affiliates, subsidiaries and their related companies for any and all claims, damages, losses and causes of action arising out of your breach or alleged breach of this agreement(disclaimer).

The materials on this site are distributed “as is” and appear on the site without express or implied warranties of any kind, except those required by the relevant legislation. In particular, I make no warranty as to the accuracy, quality, completeness or applicability of the information provided.

The information provided is for entertainment and promotional purposes only. You may not rely on any information and opinions expressed in it for any other purpose.

Disclaimer short version:

This is a DIY project, use any provided information and/or materials at your own risk! I am not responsible for what you do with it!

2. About this project

This document is related to the 14 channel Spectrum analyser. It has 14 frequency bins (channels) and each bin is built with 20 acrylic tiles and LEDS. (WS2812 Pixels).

You can connect your audio signal by using the audio input or you can use the build in microphone. Although using the microphone will limit the frequency response because of its limitations.

You can adjust input sensitivity, brightness, and peak hold time. It also has several modes that you can use to change the colour of the LEDS. When it does not receive any input signal, after a while, it will go tofire mode in which all towers will light up like a fire.

My prototype was built with acrylics and some electronics. The files to duplicate the acrylic parts are available for download but you might need to tweak them to your specific situation or design. I designed a PCB for the electronics. The PCB can be purchased at my Tindie web shop.

The firmware (Arduino Sketch) is open source and you can modify it to your needs.

3. Operation

You can use the microphone in to connect a small condenser microphone or you can connect your audio device to the line input connectors.

Mode button

A short press on the button will activate the next operation mode. Colours of the bars and peaks are different in each mode. The last mode is the fire mode. In the fire mode, the display looks like a fire. If this mode was activated by hand, it will stay in this mode until you press the mode button again.

5 short presses within 2 seconds, will activate the AutoChangeMode in which the mode will change every few seconds. (as defined by AutoChangetime in Setting.h). You know that you activated the AutoChangeMode when the display is showing the Dutch National Flag for about 2 seconds. The autoChangeMode is disabled anytime you press the mode button again.

A long press of 3 seconds will activated the hardware test routines. You'll need the serial monitor to get feedback. See the software section for more details.

Reset Button

This will reset the microcontroller. However, It will not reset the led strips.

Sensitivity Potmeter

Use this to adjust the height of the bars in relation to your input signal

Brightness Potmeter

You can use this to adjust the overall brightness of all leds / display.

Peak Delay Potmeter

You can use this to adjust the time it takes for a peak to fall down to the stack

Serial Monitor

If you enabled the DEBUG mode in the firmware of by activating and completing the hardware test, the serial monitor (USB) will output information.

4. Tools needed.

You will be working with low voltage (5V maximum) but you should know your way around basic electronics. Using the wrong voltage or polarity might not kill you but it will destroy your project in an instance!

Being successful in building this device requires some adequate skills and tools. You should be able to solder small components on a PCB which will require a soldering station with a small tip as you will be handling components like 1206 sized Although small, if you have a steady hand, you should be able to solder them onto the board. If this is beyond your possibilities, then you should ask someone to do it for you or order an already assembled PCB.

(Availability will depend on demand).

Furthermore, you be programming the microcontroller by using the Arduino IDE environment. Although the steps to do so will be described in this document, it is advisable for you to get to know this Arduino software.

If you are considering building the full-sized acrylic spectrum analyser, you will need to produce the panels and tiles for it. It can be done by hand, but it would be best to use a Laser cutter. Remember, there are some companies that offer a laser cutting service. Maybe there is a company near you who can do it for you?

And yes, you will need some basic tools like a screwdriver 😉

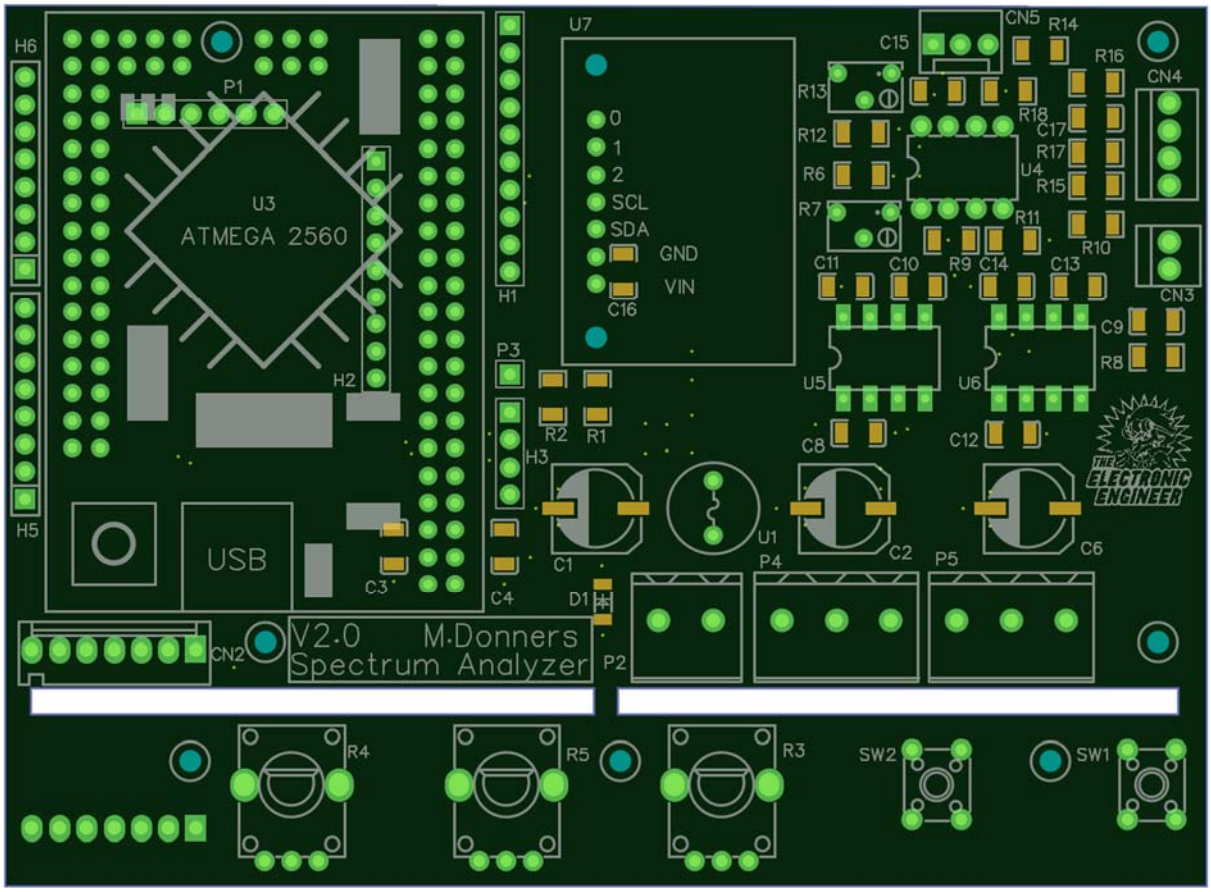
5. Hardware

5.1. Main PCB

The pcb without components is available at my Tindie store here: Tindie.

Shipping is only possible within the EU because shipping to other countries will involve import Tax, declarations of conformity and other complicated stuff. Nobody is looking for that kind of complicated formalities or import tax upon receiving the goods. If you are a resident of a country that I don't ship to, you can always order a pcb directly from a PCB supplier near you. The Gerber productions files you need for that are available for download, although it might be more expensive than what you are hoping for. If you need 1 or 2 PCB's, a Tindie order is the cheapest way to go.

5.1.1. Assembly



Assembly notes:

Start with all small SMD components. The last smd components to be mounted are C1,C2 and C6.

Use IC Sockets or Female headers to mount Ic's (U4,U5,U6)Arduino board(U3) and Frequency board(U7). Solder all the connectors last. You don't need to solder the connectors that you don't need. That's why I didn't solder H1,H2,H3,P1,P3,H5 and H6. Does connectors are only there in case I need them in the future or for debugging if needed.

Note: Although functionally speaking, the hardware V2.0 is similar and compatible with V1.0. Only 1 hardware difference: Potmeter R4 and R3 functionality are swopped!

Here are some links for some of the used components. A more detailed list of components is available on one of the following pages of this document.

Frequency Board

https://nl.aliexpress.com/item/4000040035169.html?spm=a2g0o.productlist.0.0.102638335PHmFO&algo_pvid=d718898a-b426-44d4-9594-f6773f242bf9&algo_expid=d718898a-b426-44d4-9594-f6773f242bf9-2&btsid=2100bb4716132466567397975e454d&ws_ab_test=searchweb0_0,searchweb201602_,searchweb201603_

You can discard the little gold connectors that are included. We don't need them.

Arduino
I used the Arduino ATMEGA 2560 PRO. It has to be the pro version because that one has a smaller pcb size.

https://nl.aliexpress.com/item/32909503032.html?spm=a2g0o.productlist.0.0.853064f6ZEbdnx&algo_pvid=96612a0e-f871-428f-939c-be1556028111&algo_expid=96612a0e-f871-428f-939c-be1556028111-4&btsid=2100bb4716132466035256229e454d&ws_ab_test=searchweb0_0,searchweb201602_,searchweb201603_

Microphone
A used a simple electred microphone like this one:

https://nl.aliexpress.com/item/32961327636.html?spm=a2g0o.productlist.0.0.40156749GH0jMh&algo_pvid=b3873d4e-2d66-4844-b423-45a81e07bc15&algo_expid=b3873d4e-2d66-4844-b423-45a81e07bc15-0&btsid=2100bdd516132465203027466e5419&ws_ab_test=searchweb0_0,searchweb201602_,searchweb201603_

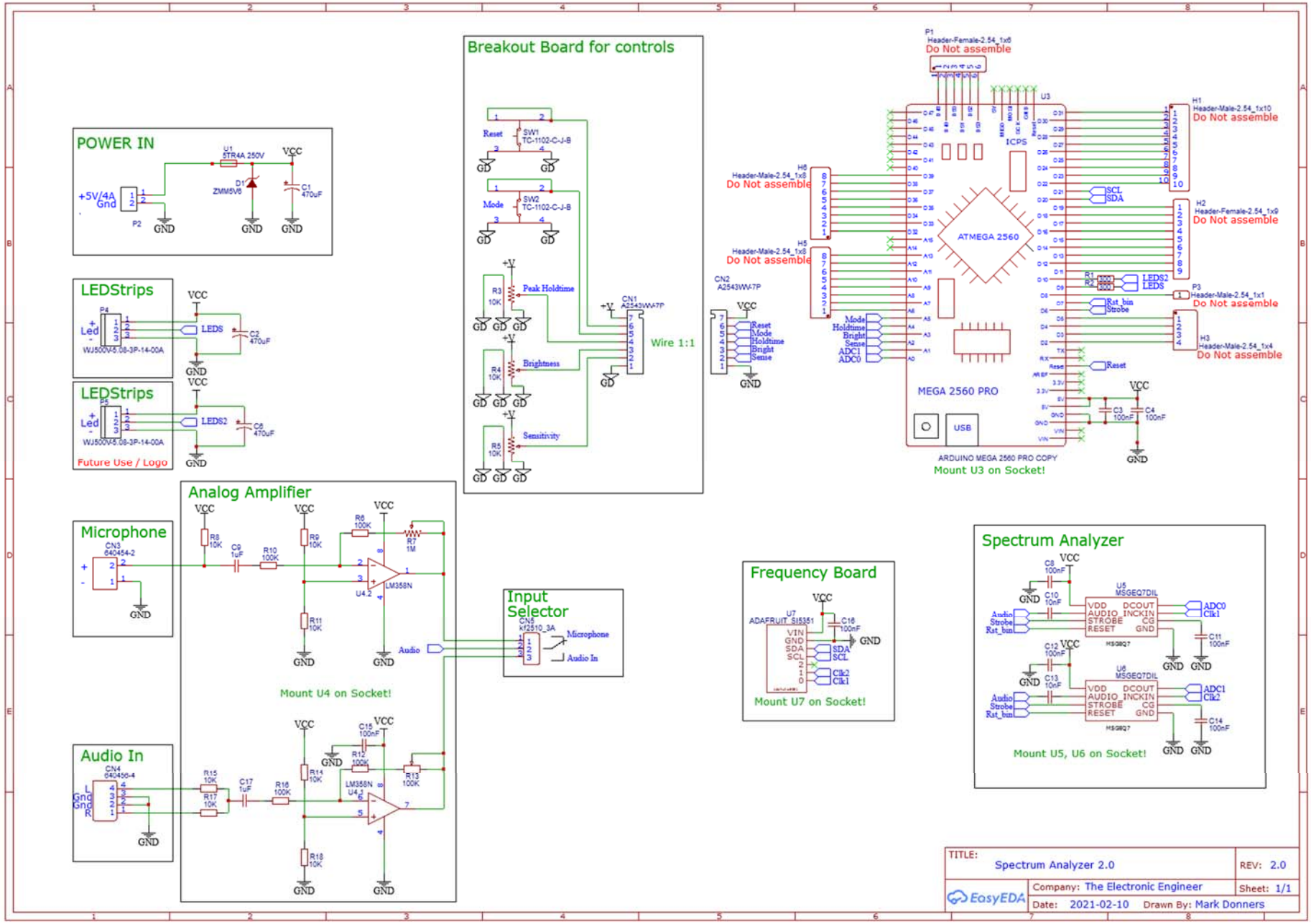
Power Supply
I used a 5V/4A power supply. Remember, my system uses 280 leds and when used at maximum brightness, even 4A will not be enough! That is why the brightness is limited in the software.

The 5V input is critical. Check the voltage of its output. It shouldn't be more then 5.1V! If your supply has a higher voltage, you could use a power diode and put it in serial with the power input line. This will create a voltage lost of about 0.6V. The board has a Zener diode over the input, if the input voltage exceeds 5.6V, this diode will start to conduct and ultimately that will result in a defective fuse. While doing a voltage test, my system work best when the input voltage was close to 5V. However, it was functioning with a input voltage between 2,9 and 5,1V. It stopped working when the voltage reached 5.17Volts. (to give you an idea how critical that 5V is!)

You'll also need some connectors for audio and power, a toggle switch to select the audio input and some wire to connect all. Regarding the power to the leds, use a wire that can handle a higher current.

Also, you'll need to order a few meter of Pixel Ledstrip WS2812. Make sure you order the one that works with your design. (number of pixels per meter varies) I used 74Leds/meter version.

5.1.2. Schematic



5.1.3. PCB Part list main PCB

The complete part list for the main component is here:

ID	Name	Designator	Footprint	Quantity	Manufacturer Part	Manufacturer	Supplier	Supplier Part
1	Connector	P2	CONN-TH_2P-P5.00_WJ500V-5.08-2P	1	WJ500V-5.08-2P-14-00A	ReliaPro	LCSC	C8465
2	470uF	C1,C2,C6	CAP-SMD_BD8.0-L8.3-W8.3-RD	3	VZH471M1CTR-0810	LELON	LCSC	C164069
3	Header-Female-2.54_1x9	H2	HDR-TH_9P-P2.54-V	1	2.54mm 1*9 Straight	BOOMELE	LCSC	C39576
4	ZMM5V6	D1	LL-34_L3.5-W1.5-RD	1	ZMM5V6	SEMTECH	LCSC	C8062
5	kf2510_3A	CN5	CONN-TH_3P-P2.00_KF2510_3A	1	kf2510_3A	BOOMELE	LCSC	C29275
6	640456-4	CN4	CONN-TH_640456-4	1	640456-4	TE Connectivity	LCSC	C86504
7	Header-Male-2.54_1x10	H1	HDR-TH_10P-P2.54-V	1	Headers Pins2.54mm1*10P		LCSC	C57369
8	100K	R13	RES-ADJ-TH_3P-L6.8-W4.6-P2.50-TL-BS-3266W	1	3266W-1-104	Chengdu Guosheng Tech	LCSC	C118925
9	WJ500V-5.08-3P-14-00A	P4,P5	CONN-TH_3P-P5.00_WJ500V-5.08-3P	2	WJ500V-5.08-3P-14-00A	ReliaPro	LCSC	C72334
10	640454-2	CN3	CONN-TH_640454-2	1	640454-2	TE Connectivity	LCSC	C86501
11	ARDUINO MEGA 2560 PRO	U3	ARDUINO MEGA 2560 PRO	1	Arduino Mega 2560 Pro Mini		Aliexpress	
12	10K	R11,R9,R8, R14,R18,R15,R17	R1206	7	1206W4F1002T5E	UniOhm	LCSC	C17902
13	Header-Male-2.54_1x1	P3	HDR-TH_1P-P2.54-V-M	1	Header-Male-2.54_1x1	ReliaPro	LCSC	C81276
14	Header-Male-2.54_1x4	H3	HDR-TH_4P-P2.54-V	1	210S-1*4P L=11.6MM	Ckmtw	LCSC	C124378
15	100K	R10,R16,R12,R6	R1206	4	1206W4F1003T5E	UniOhm	LCSC	C17900
16	10K	R4,R3,R5	RES-TH_RK09D1130C2P	3	RK09D1130C2P	ALPS Electric	LCSC	C361173
17	5TR4A 250V	U1	FUSE-TH_BD8.5-P5.08-D1.0	1	5TR4A 250V	XC Elec(Shenzhen)	LCSC	C140489
18	10nF	C13,C10	C1206	2	1206B103K500NT	FH	LCSC	C1846
19	1uF	C9,C17	C1206	2	CL31B105KBHNNNE	SAMSUNG	LCSC	C1848
20	Header-Male-2.54_1x8	H5,H6	HDR-TH_8P-P2.54-V-M	2	210S-1*8P L=11.6MM	Ckmtw	LCSC	C124381
21	100nF	C14,C11,C8, C12,C16,C3,C4,C15	C1206	8	CL31B104KBCNNNC	SAMSUNG	LCSC	C24497
22	ADAFRUIT_SI5351	U7	ADAFRUIT_SI5351	1			Aliexpress	
23	300	R1,R2	R1206	2	1206W4F3000T5E	UniOhm	LCSC	C17887
24	MSGEQ7DIL	U6,U5	DIL-08	2		Sparkfun	Mouser	
25	1M	R7	RES-ADJ-TH_3P-L6.8-W4.6-P2.50-TL-BS-3266W	1	3266W-1-105	Chengdu Guosheng Tech	LCSC	C124987
26	TC-1102-C-J-B	SW1,SW2	KEY-TH_4P-L6.0-W6.0-P4.50-LS6.5	2	TC-1102-C-J-B	XKB Enterprise	LCSC	C381016
27	A2543WV-7P	CN1,CN2	CONN-TH_PH-7P-2.54	2	A2543WV-7P	Changjiang Connectors	LCSC	C239367
28	Header-Female-2.54_1x6	P1	HDR-TH_6P-P2.54-V-F	1	Female header 1*6p	BOOMELE	LCSC	C40877
29	LM358N	U4	PDIP-8_L10.2-W5.9-P2.54-LS7.6-BL	1	LM358N	Texas Instruments	LCSC	C83405

5.2. Electronics explained

The Electronics can be divided into several sections. Of course, there is a power supply and some standard connectors for audio input and power and there are a few standard switches. Nothing scary about that at all.

The ledstrip is a OEM product that you can buy all over the planet. It is based on the WS2812 RGB LEDS. The amazing part of this setup is that all 280 LEDS are placed in serial. The output of one LED is connected to the input of the next LED. Therefore, we only need one output of the microcontroller to drive all LEDS.

A frequency board based on the SI5351 is used to generate two stable frequencies that are slightly different. These clock signals are connected to the clock input of the MSGEQ7's. So basically, instead of using the internal clock, we tell those chips what clock frequency to run on. Doing so, we are able to fool this chip to run on a slightly different frequency. It is because of this frequency shift that we can create 14 frequency bins instead of 2 times 7. Sure, you could modify the hardware slightly (adding the original clock components as mentioned in the datasheet and by leaving out the frequency board entirely) to use 2 times 7 channels-> 7 for left and 7 for right… But that would have been easy……. 😉

The PCB also includes a small dual rail to rail opamp to amplify the incoming signal. One opamp is used to boost the signal from the microphone and the other is used to boost the signal from the audio input. You will need to adjust the amplification factor by changing the trim potmeters.

The output of these opamps is connected to a switch so that you can choose one input at a time. The selected audio signal is then offered to the MSGEQ7 IC's for analysis. The microcontroller will get the frequency bins from the chips and process them accordingly.

The potmeters are used to adjust Brightness of the LEDS, Sensitivity of the inputs and falling time of the top LEDS. One switch is used to change mode and another switch is used to reset the system completely.

5.3. General

Feel free to download the AutoCAD files to 'laser cut' or mechanically engineer your own housing. It is fine if you prefer to modify that design or use your own creation instead but know that my design is freely available.

The housing is made up of two parts, the left side and the right side. Both are joined together in the middle using a transparent Acrylic plate. I also put 1 pixel led in this plate to illuminate this plate. This pixel is connected to the small Arduino that is used for the logo. I also mounted a subD9 connector male/female in this middle plate. That way it is easier to partly disassemble the unit for easy transport. The top plates of the housing are re-enforced with wooden plates inside. The acrylic led towers are mounted on the top plate and wooden re-enforcement plate using nuts.

The acrylic towers are made of 10mm transparent acryl. The 'tiles' are joined together on 2 metal rods using spacers.

I used a laser cutter to cut all the acrylic parts (5mm thickness) for this casing.

All the parts of the casing are glued together using proper glue for the used materials.

5.3.1. Assembly of the base / housing

Before you make a 1:1 copy of my housing, using the provided files, make sure that the design fits your needs.

For one, this design was made for my prototype and maybe you will use different switches, power entry etc. adjust all accordingly. I used AutoCAD to create my design.

I started gluing the sides to the top plate using glue for acrylic and I added the side's and the inside panels/re-enforcements. I also placed the wooden re-enforcement plates under the top plate. After placing this re-enforcement plate, I used my drill to make sure all the holes lined up.

You can use a metal L shape profile of about 50cm in length as a tool to make sure that you glue everything at a perfect angle. Also, I would advise you to use the glue in a less then generous way because if you use to much of it, you'll end up like me; removing the excess glue and polishing the unit to remove the stains 😉

5.3.2. Assembly of the towers

Tip: To assemble and glue everything together, a solid 90° angle will help. I use a metal profile and some clamps as support.

5.3.3. Mounting everything in its place

Once you have assembled the 14 towers, you can inserts the pixel LED strips. Don't forget to solder wires on them first. Gently insert the wires through the wiring holes in the top plate and tighten the rods using a washer and a nut for each rod. Careful! The more towers you mount, to heavier this unit will get. Make no mistake, the unit is heavy enough to brake in two if you pick it up wrong. Use two hands all the time and support the base and towers when carrying this unit around.

There is a youtube video available in which you can see how I assembled my prototype.

https://www.youtube.com/watch?v=zVXzfOgn3G0

5.4. Wiring

The wiring is not that difficult. The wiring diagram is shown on the next page.

I used shielded wire to connect the microphone and the audio input and I used some general wire for everything else.

Give some extra attention to the power lines that feed the LED Strips. You must wire the data lines in series, meaning that the data out of one strip will be connected to the data in of the next. Etc. You can also do that with the power lines, but I would not recommend that. It is better to feed each strip with its own power. This will keep the current trough the power line limited for each wire and it will better distribute the power to all leds.

5.4.1. Logo wiring

Sinds version 2.02, it is possible to connect an extra Ledstrip for the logo. In the old version this was done by using an external Arduino Tiny. This is no longer necessary. You can now hook up an extra ledstrip to connector P5. I tried the firmware with a ledstrip up to 30 leds. More might work but to many could slow down the response of the VU meter. (Although that would have to be a lot of leds….)

In the Arduino sketch the following parameters are responsible:

#define LOGO_PIN 10 // Which pin on the Arduino is connected to the Logo NeoPixels?

#define LOGOLED_COUNT 10 // how many leds in the logo

#define LogoBrightness 50 // Set BRIGHTNESS of logo

If you are using the older version hardware V1.0., P5 is not present so you would have to seek out pin 10 on the Arduino board and wire it manually.

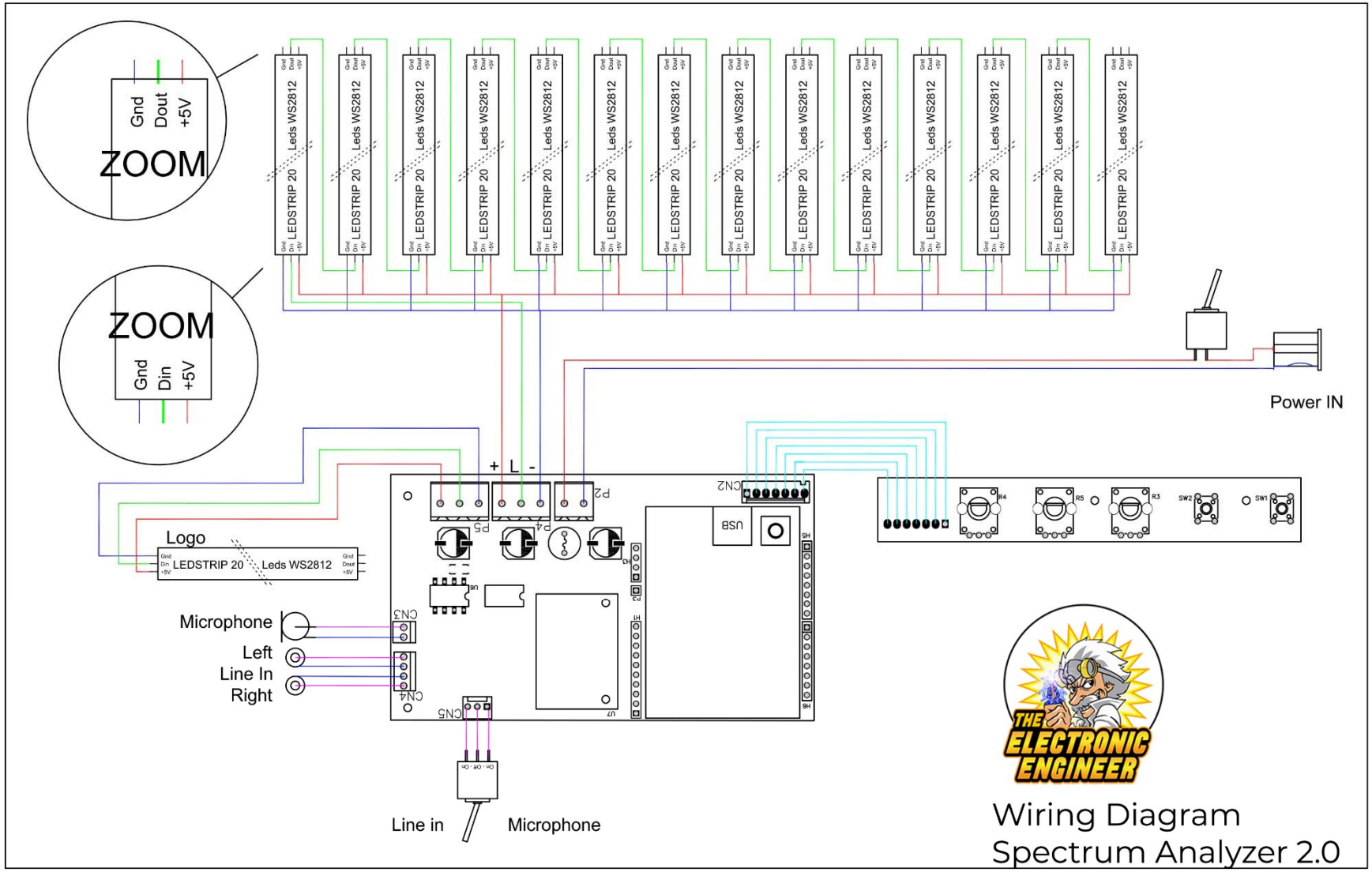
5.4.2. Main LEDSTRIPS / Matrix wiring

Note: Although the below mentioned connection for the Ledstrip/Display is default, as of firmware (sketch/software) V3.0 there is an option to redefine it if needed following the instructions of the FastLED_NeoMatrix library:

https://github.com/marcmerlin/FastLED_NeoMatrix

5.5. WiringDiagram

The wiring is not that spectacular.



6. Software

In general you can say that the program in running in repeated mode. Every loop, it will readout the data from the MSGEQ7 Ic's and store it in an Array. This array contains 14 'containers', one for each frequency band. It will then look at each container and it will translate the container value to a number of LEDS that needs to be turned on. If the number of 'new' LEDS is lower than the number of LEDS that are currently on, it will initiate a dropping of the top light. The speed it used to drop is adjustable by a variable resistor. There is also a variable resistor to adjust the brightness of the LEDS. This brightness is limited in the software because a maximum brightness of all 280 LEDS being activated at the same time, could result in a very high current. The current input is limited by a 4A fuse. The traces on the PCB are not calculated for a higher current.

There is also a variable resistor to adjust the sensibility on the input of the microcontroller's ADC inputs. The unit has two switches, one to do a reset and one to change the operating mode. (changing colours and patterns)

The actual frequency analysis is done by two external IC's: MSGEQ7. Each IC operates at a slightly different clock frequency. The clock frequency is generated by the frequency board SI5351. Because both are operating at a different frequency, the frequency analysis of the input signal is slightly shifted in the second IC. This makes it possible to create 14 bands although each IC only has 7 bands available.

6.1. Functions

The firmware can be divided into several sections:

- Definitions and Variables
- Setup
- Sub routines needed for normal operation
- Sub Routines needed for the Fire screensaver
- Sub Routines needed for Diagnostics

6.1.1. Definitions and Variables

Most variables and definitions are defined in Setting.h. Here are the most important ones:

```
// Ledstrip Logo
LOGO_PIN       10                  // second ledstrip for logo.
NUM_LEDS_LOGO  10                  // how many leds on your logo. You can define more leds then connected, that will result in wider gradient.
// Ledstrips matrix main display
LED_PIN        9                   // This is the data pin of your led matrix, or ledstrips.
COLUMNS        14                  // Number of bands on display, this is not the same as display width...because display can be 28 ( double pixels per bar)

                                   // if you have more then 16 bands, you will need to change the Led Matrix Arrays in the main file.
kMatrixWidth  14                   // Matrix width --> number of columns in your led matrix
kMatrixHeight  20                  // Matrix height --> number of leds per column
SERPENTINE     false               // Set to false if you're LEDS are connected end to end, true if  serpentine

// Ledstrips or pixelmatrix
CHIPSET        WS2812B             // LED strip type -> Same for both ledstrip outputs( Matrix and logo)
BRIGHTNESSMAX 255                  // Max brightness of the leds, carefull, to bright might draw to much amps!
COLOR_ORDER    GRB                 // If colours look wrong, play with this
LED_VOLTS      5                   // Usually 5 or 12
MAX_MILLIAMPS  2000                // Careful with the amount of power here if running off USB port, This will effect your brightnessmax.
                                      Currentlimit overrules it.
                                   // If your power supply or usb can not handle the set current, Arduino will freeze due to power drops
// ADC Filter
NOISE          20                  // Used as a crude noise filter on the adc input, values below this are ignored
//Controls
SENSITIVITYPOT  3                  // Potmeter for sensitivity input 0...5V (0-3.3V on ESP32)
BRIGHTNESSPOT   2                  // Potmeter for Brightness input 0...5V (0-3.3V on ESP32)
PEAKDELAYPOT    4                  // Potmeter for Peak Delay Time input 0...5V (0-3.3V on ESP32)
Switch1         59                 // Connect a push button to this pin to change patterns
LONG_PRESS_MS   3000               // Number of ms to count as a long press on the switch

// MSGEQ7 Connections
STROBE_PIN    6                    //MSGEQ7 strobe pin
RESET_PIN     7                    //MSGEQ7 reset pin

int BRIGHTNESSMARK= 100            // Default brightness, however, overruled by the Brightness potmeter
int AMPLITUDE    =  2000           // Depending on your audio source level, you may need to alter this value. it's controlled by the Sensitivity Potmeter
// Peak related stuff
Fallingspeed  30                   // This is the time it takes for peak tiles to fall to stack, this is not the extra time that you can add by using the
                                      potmeter for peakdelay. Because that is the extra time it levitates before falling to the stack
AutoChangetime  10                 // ISeconds between mode change, when the patterns change automatically, is too fast, you can increase this number

CRGB leds[NUM_LEDS]                // Leds on the Ledstrips/Matrix of the actual Spectrum analyzer lights.
CRGB LogoLeds[NUM_LEDS_LOGO]       // Leds on the ledstrip for the logo

NumberOfModes 13                  // The # of modes, remember it starts counting at 0, so if your last mode is 11 then the total number of modes is 12
DefaultMode 1                      // This is the mode it will start with after a reset or boot
DemoAfterSec  6000                 // if there is no input signal during this number of seconds, the unit will go to demo mode
DemoTreshold  10                   // this defines the treshold that will get the unit out of demo mode
```

Colors

There are also a whole bunch of color definitions in this file. You can change does to your likings. Take a look at the Setting.h file for more details

Fire Screensaver

The following constants are related to the Fire screensaver

```
flarerows = 8                      /* number of rows (from bottom) allowed to flare */
maxflare = 4                       /* max number of simultaneous flares */
flarechance = 50                   /* chance (%) of a new flare (if there's room) */
flaredecay = 14                    /* decay rate of flare radiation; 14 is good */
#define FPS 15                     /* Refresh rate 15 looks good*/
```

6.1.2. setup()

First the serial port is defined. Can be used for debugging
Next, the LED matrix is defined and initialized.
Next, the button is defined and initialized. It has three functions: short press, long press, and sequence press.
Then, the frequency board Si5351mcu is initialized and the output frequency of channel 0 is set to 104570Hz and the channel 1 is set to 166280Hz.
Next, the pins that need to be set as output are configures as output.
Finally, the analyser Ic's are reset.
If debug=1 in your Setting.h, the serial monitor will give you feedback during the setup function.

6.1.3. loop()

This is the main program. You can look at it as a sequence of events that always repeats itself.

1. If we are not in demo mode (fire display) the display will be cleared first.
2. The logo will be updated
3. The user inputs are scanned and processed.
4. The MSGEQ7 are addressed and all spectrum band values are stored in an array called bandValues[]
5. The data from bandValues[] is processed into band height
6. Peak heights are updated
7. If there was no input signal in the previous steps, during the time set in Setting.h, the demo mode will be activated. Before going to demo mode, the actual mode and autochange settings are stored. If the unit was already in demo mode and there was a signal detected in the previous stepts, the stored settings for mode and autochange are restored and the unit will go back to normal operation.
8. Bars and peaks are visualized. The way how, depends on the mode settings.
9. EVERY_N_MILLISECONDS(Fallingspeed) will display the peak on the right height.
10. EVERY_N_MILLISECONDS(10) The colortimer is increased by 1. This is needed for a function in some modes.
11. If the AutoChangePatterns mode is true, then the every AutoChangeTime, the mode setting will move on to the next. The fire screensaver mode is excluded.
12. Fastled will update the leds or matrix display

6.1.4. ChangeMode()

This is called after the Mode button was pressed shortly. It will re-ignite the Fastled Matrix if necessary, the AutoChangeMode will be set to False and the system will change to the next Mode. However, if this function was called while the system was in Fire(demo) mode, the stored settings for Mode are restored and the system will go back to the mode it was in before it entered the Fire mode.

6.1.5. startAutoMode()

This function is called after the mode button was pressed 5x within 2 seconds. It will store the current mode and AutoChangeMode values to memory and go the Fire mode.(demo)

6.1.6. rainbowBars(int band, int barHeight)

This is the function for displaying the leds in a rainbow pattern.

6.1.7. SameBar(int band, int barHeight)

This is the function for displaying the leds in the same color

6.1.8. SameBar2(int band, int barHeight)

See SameBar(int band, int barHeight) but maybe different color

6.1.9. TriBar(int band, int barHeight)

This is the function for displaying the leds. Each column is devided into three sections with each another color.

6.1.10. purpleBars(int band, int barHeight)

This is the function for displaying the leds in a purple gradient

6.1.11. changingBars(int band, int barHeight)

This is the function for displaying the leds in a changing rainbow pattern

6.1.12. centerBars(int band, int barHeight)

This is the function for displaying the leds. Origin is in the centre of each column, moving to top and bottom, following a fixed color pattern

6.1.13. centerBars2(int band, int barHeight)

See centerBars(int band, int barHeight)

6.1.14. centerBars3(int band, int barHeight)

See centerBars(int band, int barHeight)

6.1.15. NormalPeak(int band, int H, int S, int V)

This is the function for displaying the peaks in a fixed color, as defined in H,S and V

6.1.16. TriPeak(int band)

This is the function for displaying the peaks, following the triband pattern where the color of the peak depends on the location of the peak.

6.1.17. TriPeak2(int band)

See TriPeak(int band)

6.1.18. outrunPeak(int band)

This is the function for displaying the peaks, where the color of the peak depends on the location of the peak. Colors are defind in a color Palette

6.1.19. Subrountines for Fire Screensaver

See the information of the original MatrixFireFast sketch by Patrick Rigney
Github: https://github.com/toggledbits/MatrixFireFast

6.1.20. Run_Diagnostics()

The device has a diagnostic mode in which you can test most off it functions. The debug mode will send back information over the serial (USB) port so you can use the Arduino Serial monitor to see what's going on.

To enter the diagnostic mode you need to press and hold the mode key for 5 seconds

```
15:46:17.798 -> Debug: ********************************************************************************
15:46:17.798 -> Debug: *   Diagnostic Mode                                                            *
15:46:17.798 -> Debug: ********************************************************************************
15:46:17.852 -> Debug: *    Arduino Sketch Version 3.00                                               *
15:46:17.852 -> Debug: *    Mark Donners, The Electronic Engineer                                     *
15:46:17.852 -> Debug: *    Website:   www.theelectronicengineer.nl                                   *
```

```
15:46:17.852 -> Debug: *    facebook:  https://www.facebook.com/TheelectronicEngineer                    *
15:46:17.852 -> Debug: *    youtube:   https://www.youtube.com/channel/UCm5wy-2RoXGjG2F9wpDFF3w          *
15:46:17.852 -> Debug: *    github:    https://github.com/donnersm                                      *
15:46:17.899 -> Debug: ********************************************************************************
15:46:17.899 -> Debug:
15:46:17.899 ->
15:46:17.899 -> Debug: ********************************************************************************
15:46:17.899 -> Debug: * The Colors of the Dutch Flag will now alternate on all leds                     *
15:46:17.899 -> Debug: * Press the Mode button to exit                                                  *
15:46:17.899 -> Debug: ********************************************************************************
```

At this point all the Pixel leds of your main display/ledstrips will light up, showing the Dutch National Flag.

- Press the mode button again and the following lines will be added:

```
15:47:18.775 -> Debug: ********************************************************************************
15:47:18.775 -> Debug: * Now showing Rainbow mode                                                       *
15:47:18.775 -> Debug: * Press the Mode button to exit                                                  *
15:47:18.775 -> Debug: ********************************************************************************
```

At this point all leds will light up like a rainbow.

- Press the mode button again and the following lines will be added:

```
15:48:14.449 -> Debug: ********************************************************************************
15:48:14.503 -> Debug: * Now Only the Logo Ledstrip will blink 9x in red color 1 sec on/ 1 sec off          *
15:48:16.516 -> Debug: * Logo Blink test 1 of 9 done                                                    *
15:48:18.512 -> Debug: * Logo Blink test 2 of 9 done                                                    *
15:48:20.563 -> Debug: * Logo Blink test 3 of 9 done                                                    *
15:48:22.553 -> Debug: * Logo Blink test 4 of 9 done                                                    *
15:48:24.573 -> Debug: * Logo Blink test 5 of 9 done                                                    *
15:48:26.616 -> Debug: * Logo Blink test 6 of 9 done                                                    *
15:48:28.634 -> Debug: * Logo Blink test 7 of 9 done                                                    *
15:48:30.638 -> Debug: * Logo Blink test 8 of 9 done                                                    *
15:48:32.689 -> Debug: * Logo Blink test 9 of 9 done                                                    *
15:48:32.689 -> Debug: * Testing of Logo Ledstrip is done                                               *
15:48:32.689 -> Debug: ********************************************************************************
```

- Press the mode button again and the following lines will be added:

```
15:49:26.404 -> Debug: ************************** Diagnostic LED Test Finished  **************************
15:49:26.457 -> Debug: * You where able to see the red, white and blue Flag? It's the Dutch Flag!          *
15:49:26.457 -> Debug: * All leds where on? No defective ones? Also, the Logo was blinking 9x in red, right?*
15:49:26.457 -> Debug: * Press the button again to continue                                              *
15:49:26.457 -> Debug: ********************************************************************************
```

- Press the mode button again and the following lines will be added:

```
15:50:18.453 -> Debug: ********************** Frequency Board Test ***************************************
15:50:18.453 -> Debug: * To test the outputs of the frequency board Remove both MSGEQ7 Ic's from the socket *
15:50:18.490 -> Debug: * place a 1K resistor in each socket between pin 3(output) and pin 8(clock)         *
15:50:18.490 -> Debug: * Channel 0 should output a frequency around 5 Khz. while channel 1 will give 10Khz  *
15:50:18.490 -> Debug: * This is not accurate measurement and only a indication. Value +/- 500Hz is fine    *
15:50:18.490 -> Debug: * If a channel gives you a measurement of 0, it means it is not working            *
15:50:18.490 -> Debug: * To exit, press and hold the mode key for 3 seconds                              *
15:50:18.537 -> Debug: ********************** Frequency Board Test ***************************************
15:50:18.537 -> Debug:
15:50:18.537 ->
15:50:21.320 -> Debug: Measured frequency channel 1: 0 Hz  channel 2: 0 Hz
15:50:24.581 -> Debug: Measured frequency channel 1: 0 Hz  channel 2: 0 Hz
15:50:27.835 -> Debug: Measured frequency channel 1: 0 Hz  channel 2: 0 Hz
15:50:31.094 -> Debug: Measured frequency channel 1: 0 Hz  channel 2: 0 Hz
```

Lines will be added while in this mode. You can see the actual signal frequency of channel 1 and 2. The measured frequency is an estimation and not very accurate. However, it will tell you if both channels of the frequency board are

working. For this test, you need to remove the MSGEQ7 Ic's from their socket. (if you haven't already. Don't forget to turn off the power and to disconnect the USB connector when you remove them). Now, for each socket, you have to use a resistor (min 1K , max 100K) connect the resistor between pin 8(Clock) and pin 3(output) of each socket. This resistor will forward both clockpulses of the frequency board to each ADC channel of the arduino. When connected corectly, and with a working frequency board it will give the following (approximitly) values:

```
19:51:09.439 -> Debug: Measured frequency channel 1: 5000 Hz  channel 2: 9803 Hz
```

This is not accurate measurement but channel 1 wil be around 5000Hz and channel 2 around 10000Hz.

If you see a value of 0 then that channel is not active (defective).

- Press the mode button and hold for 3 seconds to exit. The following will be added:

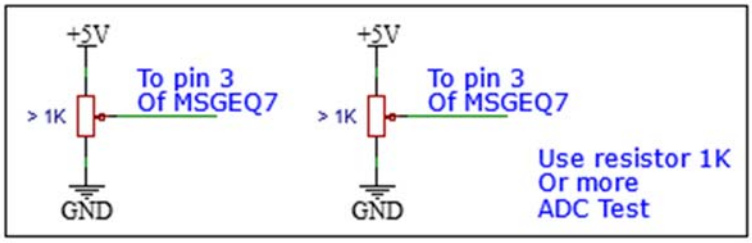
```
15:53:19.506 -> Debug: ****************************** Amplifier test ************************************
15:53:19.506 -> Debug: * This will print both adc values until you press the mode button.                  *
15:53:19.506 -> Debug: * Press the mode button to begin                                                    *
15:53:19.506 -> Debug: ****************************** Amplifier test ************************************
```

- Press the mode button to begin. A loop will continue the following line:

```
19:57:25.659 -> Debug: ADC value 0:  127   ADC Value 1:  128
```

It shows the direct digital value of the analog channel 0 and 1. With this you can test the ADC converter of the Arduino. You can apply a voltage on pin 3 of the MCGEQ7 sockets to do the test. (MSGEQ7 are removed from socket)

The easiest way to test is to connect a variable resistor (potmeter) to the pins. Like so:



You can now turn each potmeter while looking at those ADC values. They should change when you turn the potmeter. (0..1023)

If you are only using 1 potmeter at a time, testing only one adc at a time, do not forget to ground the channel that you are not using by adding a wire or resistor between pin 2 and 3. This is necessary because interference will make it look like you are applying the test signal to both adc inputs.

- Press the mode button to exit the ADC test. The following text will be added:

```
15:54:36.722 -> Debug: ****************************** Potmeter test ************************************
15:54:36.722 -> Debug: * This will print the mapped values potmeters until you press the mode button       *
15:54:36.722 -> Debug: * Sense: 50-1023 , Brightness: 10-Brightnessmax , Peak Delay: 1-150                 *
15:54:36.722 -> Debug: * Press the mode button to begin                                                    *
15:54:36.769 -> Debug: ****************************** Potmeter test ************************************
```

- Press the mode button to exit the test. The following text will be added:

```
15:55:52.296 -> Debug: Sense (50-1023):  852  -  Brightness(10-Brightnessmax):  213  -  Peak Delay(1-150): 125
```

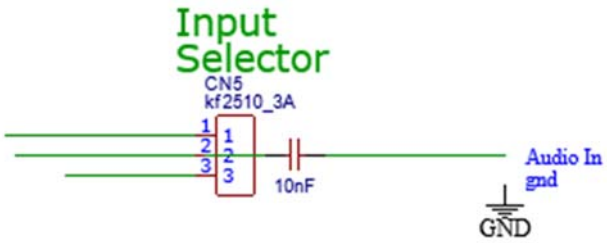
- Press the mode button to exit the test. The following text will be added:

```
15:55:53.847 -> Debug: * When you press the mode button, the system will go to normal operation mode but   *
15:55:53.847 -> Debug: * with the debug feedback on.                                                        *
15:55:53.847 -> Debug: ******************************** END of TEST ************************************
```

- Press the mode button to exit to return the system to normal mode. However, the debug mode will stay on until you reset the system. During normal operation and while in the debug mode, feedback is given to the serial monitor.

Opamp test not included

Testing the Opamp is not done by the software. However, if you think that the opamp is causing you trouble, you can bypass it completly by removing the input selector and applying the audio signal directly to the center pin of this connector (CN5) using a capacitor of 10 nf. Like so:



7. Programming your Arduino

I used the Arduino IDE. It is freely available online and it does the job. However, I recently stumbled on something called Sloeber Beryllium which is a great tool that offers a much better compiler interface. However, it has a bit of a learning curve but I promise, it's worth it! Why don't you check it out? You can also use Visual Studio or some other great IDE. However, it is important the right library and it is best not to install what you don't need as it might give you errors when compiling.

7.1. Here a few libraries that you'll need for sure:

Using library FastLED_NeoMatrix at version 1.1 in folder:
C:\Users\chord\Documents\Arduino\libraries\FastLED_NeoMatrix
(Just find it in the Arduino IDE libraries)

Using library Adafruit_GFX_Library at version 1.10.4 in folder:
C:\Users\chord\Documents\Arduino\libraries\Adafruit_GFX_Library
(Just find it in the Arduino IDE libraries)

Using library FastLED at version 3.4.0 in folder: C:\Users\chord\Documents\Arduino\libraries\FastLED
(Just find it in the Arduino IDE libraries)

Using library EasyButton at version 2.0.1 in folder:
C:\Users\chord\Documents\Arduino\libraries\EasyButton
(Just find it in the Arduino IDE libraries)

Using library Si5351mcu-master at version 0.6.2 in folder:
C:\Users\chord\Documents\Arduino\libraries\Si5351mcu-master
You can download it here: https://github.com/donnersm/Si5351mcu

Remark: I had some trouble compiling when I started. Turned out that Arduino IDE had many libraries activated and it decided to choose the wrong ones whenever it had to choose between libraries. I solved it by uninstalling the Arduino IDE and re-installing it from scratch.

8. Trouble shooting

If the device is not working, there are several things you can check.

First Column(left), first few leds flickering.....

This is low frequency noise coming from the powersupply and it is amplified by the opamp.
I Compensated a little for it in the software but depending on your Power supply, it might still be there. In my prototype it only happens when I use the mic input. the line-in is working just fine. If you have trouble with this, you can try changing opamp to a different type or change the power supply, or add a high pass filter , blocking the low noise but that's not easy since it is related to the net frequency of 50 / 60 hz.
You can compensate for it in software.

It's not working

If nothing is working or parts are not doing what you expect, remember the system can be divided into function blocks that you can test one by one:

1. **Check your power supply is is 5V?**
   When you connect the input to ground so that it will not pickup noise, after a few seconds or so, the unit goes into fire mode. This is seen as a colorful set of bars displaying a fire. If this is working then you know that the software is running and the driving of the LEDS works fine!
2. **I had some feedback that some units work on USB power but not on external power.**
   Check the voltage of both USB and external supply. I most cased, the external supply was 0.5V higher then the USb voltage. You can create a voltage drop by placing a universal diode in series with the +5V input, coming from the external supply. This will create a drop of around 0.6V. Make sure the diode can handle 4A current. (I used a few 1N4001 in parallel)
3. **No leds turn on ever!**
   check your connections to the ledstrip. Is it wired correctly?
   When you remove the msgeq7 Ic's from their socket, you can apply a voltage to pin 3 of the socket, depending on this voltage(0-3 V) , you should see 7 bars increasing in numbers of leds that turn on, related to the voltage. You can test this for both the MSGEQ7 Ic's. If the leds turn on then you problem is with the MSQEQ7 or the preamp. You can leave out the preamp and connect your audio directly to the center pin of the jumper(Selection switch), bypassing the preamp.
4. **Did you hook up the potmeters?**
   One is for sensitivity and one is for brightness. You'll need to install them correctly!
5. **Did you connect the selection switch and is it in the correct position?**
   You need to connect this switch to choose what input you want to use.
6. **Is the frequency board working?**
   You should see a clock signal on pin 8 of both MSGEQ7 but you'll need an oscilloscope to check it. Most likely, if everything checks out but it is still not working, the MSGEQ7 is the problem. There are many fakes out there, specially china import....dont trust the MSGEQ7 from Alieexpress. Even my local supplier got me not-working ones.....in the end I ordered them at Mouser. Mouser sells directly from Sparkfun who is the manufacturer these days... You can check the function of this IC to makek sure it's working. If it has a clock and an audio input and a strobe, you should see the output changing all the time. It kinda looks like stairs but the steps are always changing.
7. **Are you looking at the right in/output?**
   The pins in the arduino schematic might be different then the pin numbers in the arduino sketch. You can use the arduino MEGA 2560 PRO pinout chart to match the number with the pin name's!!