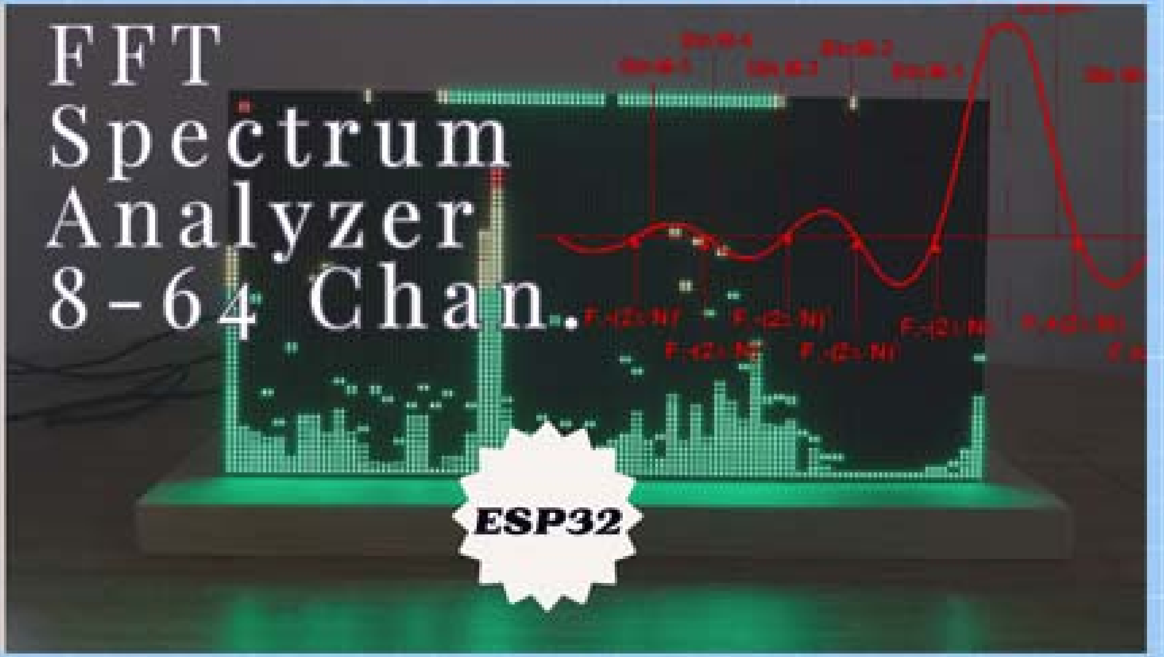
8-64 CHANNEL ESP32 SPECTRUM ANALYZER



BUILDING MANUAL

M. Donners 01-06-2021

Table of Contents

1. Disclaimer and safety 3
2. About this project 3
3. Operation 4
4. Tools needed. 5
5. Hardware 6
    5.1. Main PCB 6
        5.1.1. Assembly 6
        5.1.2. Schematic 8
        5.1.3. PCB Part list main PCB 9
    5.2. Electronics explained 10
    5.3. Visualisation 11
        5.3.1. Using one or more HUB75E Panels 11
        5.3.2. Using a Pixeled Matrix (WX2812 based or similar) 11
    5.4. Wiring 12
        5.4.1. HUB75E display wiring 12
        5.4.2. Main LEDSTRIPS / Matrix wiring 12
    5.5. Wiring tables 13
6. Software 14
    6.1. Files 14
        6.1.1. Settings 15
7. Programming your Arduino 17
    7.1. Here a few libraries that you'll need for sure: 18
8. Trouble shooting 19

M Donners Document version 1.0 July 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 FFT Spectrum analyser. It has 8, 16,24,32 or even 64 frequency bins (channels) and you might be able to double that number of bins if you modify the firmware.

The pcb is capable of driving a matrix of pixelleds (WS2812 or led matrix) or you can connect one or more HUB75E displays but you will have to make a choice which one to use and adjust the parameters in the settings accordingly.

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

The input sensitivity is automated and it will adjust to the levels off the input. You can adjust brightness and peak hold time. When it does not receive any input signal, after a while, it will go to fire mode in which some leds/display will light up like a fire.

I used two HUB75E panels for my prototype as I mounted them on a wooden stand. The PCB was mounted to the back of the display.   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

The mode button has 3 functions:

Short press: change pattern(mode), there are 12 available patterns from which the last one is a fire screensaver.

Fast double press: Change the calibration filter. You can choose from none, white, pink or brown noise. Each mode will manipulate the output data accordingly.

For white noise, a white noise sound source of 1Vpp was connected to the input and all the output values where measured. The measured values where recalculated and copied to the filter database. When calibrated, all the output values are almost equil in amplitude when the mentioned source is connected. I did the same for pink and brown noise.

Long press: Press and hold for a few seconds to enable/disable the automode. When automode is enabled, the patterns will change every few seconds.

Reset Button

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

Brightness Potmeter

You can use this to adjust the overall brightness of all leds / display. WARNING:Make sure you use a power supply to match the current for the brightness that you set. For sure, the ESP32 onboard regulator can not handle all leds at full brightness. It is best to use an external powersupply that can handle 4 to 6 A. If you are using the USB cable that is connected to the ESP32, you might end up with a burning sensation coming from the ESP32 Board.

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 and 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 0805 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 buy the pcb version that has all SMD components pre-installed.

Furthermore, you will 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.

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

5. Hardware

5.1. Main PCB

The pcb is available at my Tindie store here:

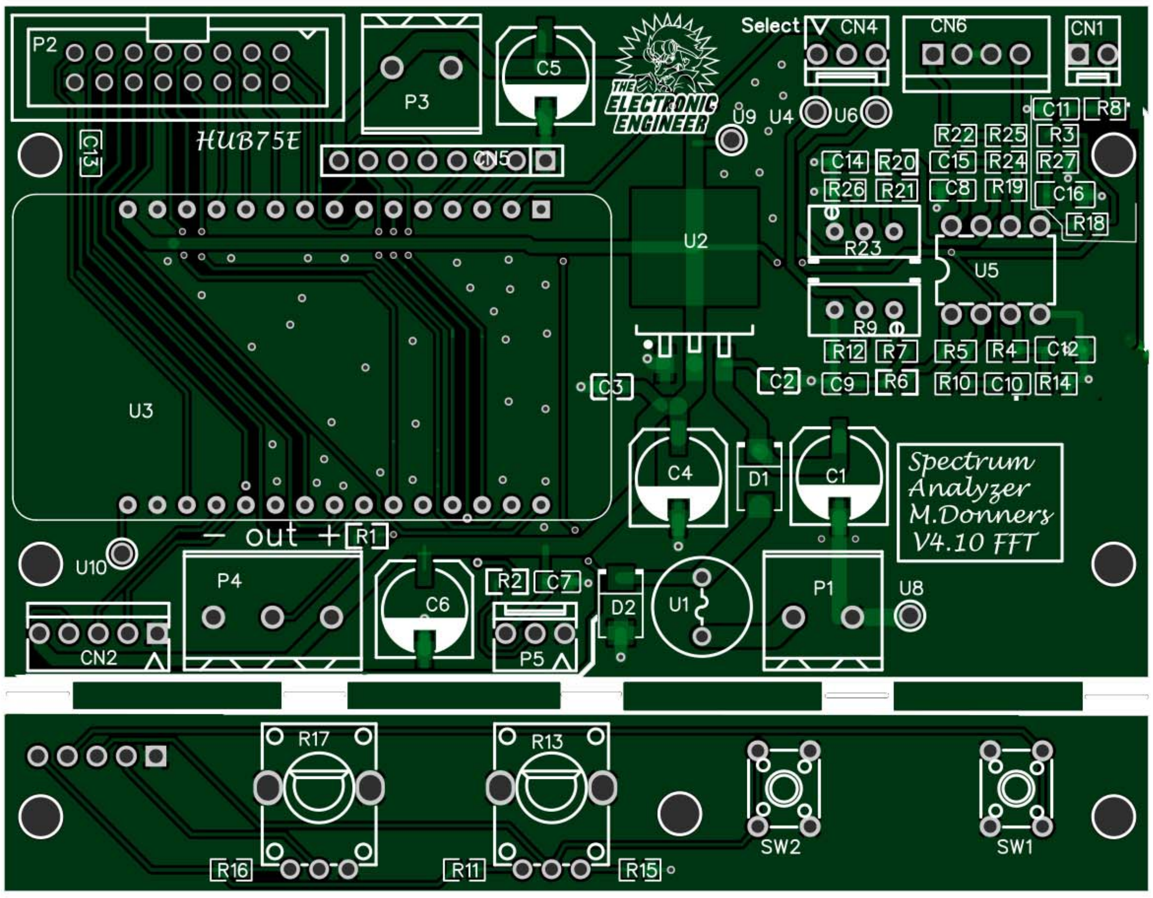
https://www.tindie.com/products/markdonners/pcb-8-64-channel-fft-spectrum-analyzer/

The PCB has all SMD components pre-installed and all you nee to add is the ESP32 board and a hand full of small components.

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. I do make some exceptions for USA and UK. 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.

My shipping policy might change over time so always check the Tindie store to see if I am shipping to your country. ADDITION: I have successfully shipped to UK, USA and other non-EU countries but handling the tax, import fee and so on is on you! Check out the Tindie store to see what is possible for your country.

5.1.1. Assembly



Assembly notes:

Start with all small SMD components. The last smd components to be mounted are C1,C4,C5 and C6.

Use IC Sockets or Female headers to mount Ic's (U5), ESP32 board (U3).It is also wise to use a socket for the fuse (U1). Solder all the connectors last. You don't need to solder the connectors that you don't need. Connector CN5 is only needed if you are planning to use those extra pins but if not, you can leave it out. Connectors P2 and P3 are to be used to connect a HUB75E panel while connector P4 is used for connecting ledstrips or a pixel-led based matrix. You can decide to leave out the connectors you do not need.

C13 has been added to compensate Latency Timing for some versions of the board I run into in a shipment from May 2021.

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.

Microphone
A used a simple electret 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

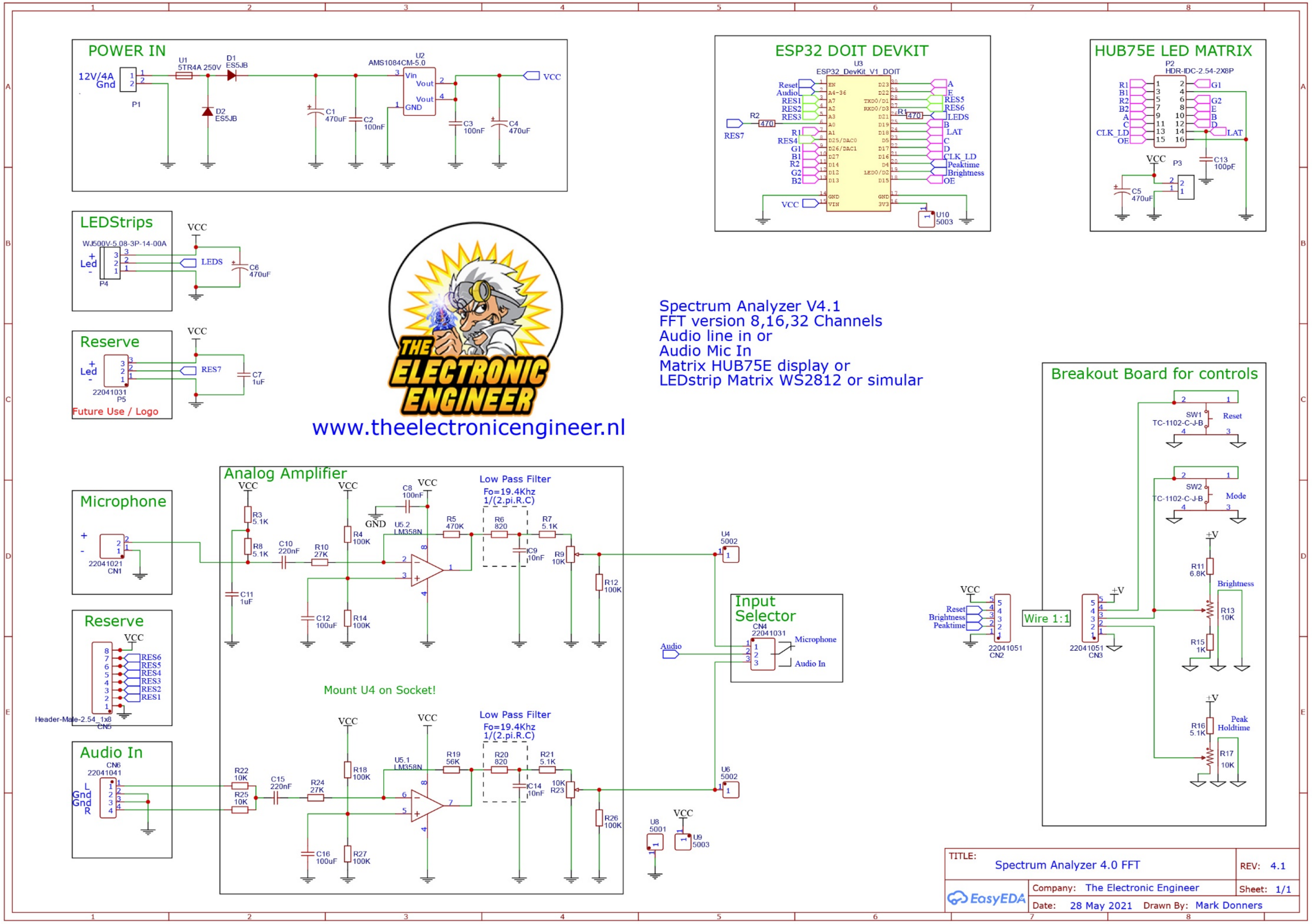
You will be needing a power supply of DC 12V that can handle 4 to 6A. (Current depends on the number of panels or leds you are planning to connect. Make sure your power supply is stable and doesn't have noise on the output.

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 current of 4A Depending on what you planning, thinner wire can do the job as well.

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.

Or are you planning to use one or more HUB75 led panels? I used 2 64*64 panels to create a matrix of 64 pixels height and 128 pixels width.

5.1.2. Schematic



5.1.3. PCB Part list main PCB

The complete part list for the main PCB is here:

ID	Name	Designator	Footprint	Quantity	Manufacturer Part	Manufacturer	Supplier	Supplier Part
1	470uF	C1	CAP-SMD_BD8.0-L8.3-W8.3-RD	1	VZH471M1CTR-0810	LELON	LCSC	C164069
2	100nF	C2,C3	C0805	2	CC0805KRX7R9BB104	YAGEO	LCSC	C49678
3	470uF	C4,C5,C6	CAP-SMD_BD8.0-L8.3-W8.3-RD	3	VZH471M1CTR-0810	LELON	LCSC	C164069
4	100nF	C8	C0805	1	CC0805KRX7R9BB104	YAGEO	LCSC	C49678
5	1uF	C7,C11	C0805	2	CL21B105KBFNNNE	SAMSUNG	LCSC	C28323
6	10nF	C9,C14	C0805	2	CL21B103KBANNNC	SAMSUNG	LCSC	C1710
7	220nF	C10,C15	C0805	2	CL21B224KBFNNNE	SAMSUNG	LCSC	C5378
8	100uF	C12,C16	C1206	2	1206X107M6R3NT	FH	LCSC	C178304
9	100pf	C13	C0805	1	CL21C101JBANNNC	SAMSUNG	LCSC	C1790
10	22041021	CN1	CONN-TH_22041021	1	22041021	MOLEX	LCSC	C185215
11	22041051	CN2,CN3	CONN-TH_22041051	2	22041051	MOLEX	LCSC	C505160
12	22041031	CN4,P5	CONN-TH_3P-P2.54_22041031	2	22041031	MOLEX	LCSC	C293437
13	Header-Male-2.54_1x8	CN5	HDR-TH_8P-P2.54-V-M	1	210S-1*8P L=11.6MMGold-plated black	Ckmtw	LCSC	C124381
14	22041041	CN6	CONN-TH_22041041	1	22041041	MOLEX	LCSC	C185196
15	ES5JB	D1,D2	SMB_L4.6-W3.6-LS5.3-RD	2	ES5JB	Shandong Jingdao Microelectronics	LCSC	C123908
16		P1,P3		2	WJ500V-5.08-2P-14-00A	ReliaPro	LCSC	C8465
17	HDR-IDC-2.54-2X8P	P2	IDC-TH_16P-P2.54_C3406	1	IDC Box 2.54mm 16P Straight	BOOMELE	LCSC	C3406
18	WJ500V-5.08-3P-14-00A	P4	CONN-TH_3P-P5.00_WJ500V-5.08-3P	1	WJ500V-5.08-3P-14-00A	ReliaPro	LCSC	C72334
19	470	R1,R2	R0805	2	0805W8F4700T5E	UniOhm	LCSC	C17710
20	470K	R5	R0805	1	0805W8F4703T5E	UniOhm	LCSC	C17709
21	6.8K	R11	R0805	1	0805W8F6801T5E	UniOhm	LCSC	C17772
22	10K	R13,R17	RES-TH_RK09D1130C2P	2	RK09D1130C2P	ALPS Electric	LCSC	C361173
23	1K	R15	R0805	1	0805W8F1001T5E	UniOhm	LCSC	C17513
24	5.1K	R3,R7,R8,R16,R21	R0805	5	0805W8F5101T5E	UniOhm	LCSC	C27834
25	56K	R19	R0805	1	0805W8F5602T5E	UniOhm	LCSC	C17756
26	820	R6,R20	R0805	2	AC0805FR-07820RL	YAGEO	LCSC	C229015
27	10K	R22,R25	R0805	2	RNCS0805BKE10K0	Stackpole Elec	LCSC	C346681
28	10K	R9,R23	RES-ADJ-TH_3296W	2	3296W-1-103LF	BOURNS	LCSC	C34846
29	27K	R10,R24	R0805	2	0805W8F2702T5E	UniOhm	LCSC	C17593
30	100K	R4,R12,R14,R18,R26,R27	R0805	6	0805W8F1003T5E	UniOhm	LCSC	C17407
31	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
32	5TR4A 250V	U1	FUSE-TH_BD8.5-P5.08-D1.0	1	5TR4A 250V	XC Elec(Shenzhen)	LCSC	C140489
33	AMS1084CM-5.0	U2	TO-263-3_L8.6-W10.2-P2.54-LS14.4-TL	1	AMS1084CM-5.0	AMS	LCSC	C56105
34	ESP32_DevKit_V1_DOIT	U3	ESP32_DEVKIT_V1_DOIT	1				
35	5002	U4,U6	TEST-TH_BD2.54-P1.39	2	5002	Keystone	LCSC	C238123
36	LM358N	U5	PDIP-8_L10.2-W5.9-P2.54-LS7.6-BL	1	LM358N	Texas Instruments	LCSC	C83405
37	5001	U8	TEST-TH_BD2.54-P1.39	1	5001	Keystone	LCSC	C238122
38	5003	U9,U10	TEST-TH_BD2.54-P1.39	2	5003	Keystone	LCSC	C238124
REMEMBER: MOST COMPONENTS ARE STANDARD COMPONENTS AND CAN BE REPLACED BY OTHER TYPE/ BRAND ETC. USE THIS LIST AS A GUIDE NOT AS AN ABSOLUTE MUST!								

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.

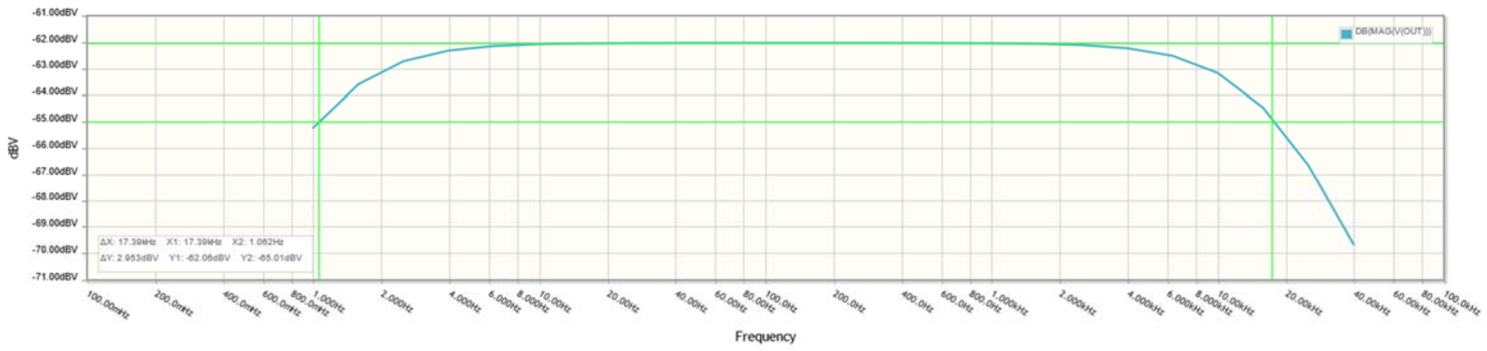
Connector P1 is used to connect a 12V power supply. The circuit is protected by a fuse and two power diodes. If the power is connected reversed, D2 will conduct so that the current will rise beyond the capabilities of the fuse. While D1 will protect the circuit from this reversed voltage, the fuse will blow. However, under normal circumstances, the power source is connected correctly and while passing a few capacitor to smooth out any power supply noise or instabilities, the 12V is offered to the on board regulator U2. U2 converts the 12V power to 5V and feeds the ESP32 board and some of our onboard features like the pre-amplifier, the led strip or matrix display.

The user interface consists of 2 potentiometers and two switches. The potmeters control brightness and Peak time delay while the switches are used to set mode, filter and auto-change mode or to give a system reset. To save out on IO on the ESP32, switch SW2 and potmeter R13 are combined in on circuit. When ever the switched is pressed, the voltage on this I/O line will be 0V (gnd) but when not pressed, the voltage on the I/O line will be between 0.3V and 3V, depending on the setting of the potmeter. This makes it possible to use 1 I/O input for both the potmeter and switch at the same time.

The PCB has room for a dual pre-amplifier. One can be used to connect a microphone and the other can be used for the line in. You have to set a jumper or use a toggle switch connected to CN4 to select which input you want to use.

Both amplifier use a low pass filter to keep the upper frequencies away from the microcontroller. Except from the microphone offset that is created with R3 and R8, both circuits are the same. Below is a diagram of the frequency response for both circuits. Although the frequency response of the amplifier looks promising, the microphone will be the frequency limiting factor. That is why you can expect better frequency bandwidth from the line input.

In regards to the HUB75 panels that are connected to P2, C3 has an important function. In May 2021, I received a batch on HUB75E panels that uses a mix of chipsets and we had a lot of trouble to get them to respond properly without glitches. Without C3 this was nearly impossible. C3 is connected to the LATENCY pin and slows down the flanks of the LATENCY Pulse. Also, the length and quality of the flatcable, used to connect the panel to the pcb should be as short as possible! C3 is 100pF in my setup but, depending on the length of the flatcable, you might want to change that. However, maybe you have more luck than me and you'll receive a whole different batch that has no problem whatsoever.



-3db@ 1Hz and 20Khz

5.3. Visualisation

You can select 1 of 2 option for visualizing the spectrum data. You have to make a selection in the settings panel of the Arduino sketch to match your setup. You can use WS2812 or similar ledstrips/Matrix or you can use HUB75 displays.

LEDDRIVER.H

```
// select one of these and comment out the other
//#define Ledstrip
#define HUB75
```

5.3.1. Using one or more HUB75E Panels

You can connect several panels together to increase the height or width of your matrix. In my prototype, I used two 64x64 HUB75E panels. So the height of my matrix was 64 pixels and the width was 128. Depending on the type of HUB75 display you are using, you might have to change some of the parameters to get it to work correctly. Also note that it is not likely that two panels with different chipset will work together. If you need more info on how and what, regarding these panels, take a look at the github of the driver, all info (and driver) is here:

mrfaptastic/ESP32-HUB75-MatrixPanel-I2S-DMA: An Adafruit GFX Compatible Library for the ESP32 to drive 64x32px or 64x64px HUB75 LED matrix modules using the ESP32 DMA Engine for faster refresh rates. Supports panel chaining. (github.com)

The following settings apply to the use of this panel:

LEDDRIVER.H

```
// select one of these and comment out the other
//#define Ledstrip
#define HUB75
#define PANEL_WIDTH     64
#define PANEL_HEIGHT    64          // Panel height of 64 will required PIN_E to be defined.
#define PANELS_NUMBER 2             // Number of chained panels, if just a single panel, obviously set to 1
```

5.3.2. Using a Pixeled Matrix (WX2812 based or similar)

The software was tested with two Led Matrixes that where connected together. Each matrix has a dimension of 16x16 pixels so the led matrix I tested with was 16 pixels in height and 32 pixels wide. Remember that the more leds you use, the slower the matrix will be. If the matrix is too big, it will react visibly slow.

You can tell the firmware how your ledmatrix is configured by changing the following parameters

see https://github.com/marcmerlin/FastLED_NeoMatrix for Tiled Matrixes, Zig-Zag and so forth

LEDDRIVER.H

```
FastLED_NeoMatrix *matrix = new FastLED_NeoMatrix(leds, kMatrixWidth, kMatrixHeight,
                    NEO_MATRIX_BOTTOM     + NEO_MATRIX_RIGHT +
                    NEO_MATRIX_COLUMNS    + NEO_MATRIX_ZIGZAG +
                    NEO_TILE_TOP + NEO_TILE_LEFT + NEO_TILE_ROWS);
```

If your ledstrip are lighting up in the wrong direction or if there is a difference between odd and even rows, this is these are the settings you need to adjust.

If you are using this PCB to replace the older PCB design with frequency board and MSGEQ7's and you see that the directions of the ledstrips is zigzag and left /right side are exchanged, try this:

LEDDRIVER.H

```
FastLED_NeoMatrix *matrix = new FastLED_NeoMatrix(leds, kMatrixWidth, kMatrixHeight,
NEO_MATRIX_BOTTOM       + NEO_MATRIX_LEFT +
NEO_MATRIX_COLUMNS      + NEO_MATRIX_PROGRESSIVE +
NEO_TILE_TOP + NEO_TILE_LEFT + NEO_TILE_ROWS);
```

Some more settings that effect the use of ledstrips are the following:

LEDDRIVER.H

```
// Ledstrip settings
#define CHIPSET                     WS2812B      // LED strip type
#define LED_PIN                     21           // LED strip data Pin, only change if you use different hardware
#define SERPENTINE                  false        // Set to false if you're LEDS are connected end to end, true if serpentine
#define COLOR_ORDER                 GRB          // If colours look wrong, play with this
#define LED_VOLTS                   5            // Usually 5 or 12
#define MAX_MILLIAMPS               2000         // Careful with the amount of power here if running off USB port
const uint8_t kMatrixWidth =        32;          // Matrix width --> number of columns in your led matrix
const uint8_t kMatrixHeight =       16;          // Matrix height --> number of leds per colum
```

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. HUB75E display wiring

If you are using HUB75 Panels, make sure to use a decent flatcable and keep it's length as short as possible. Simple connect the flatcable to the PCB and display and don't forget to connect the powerline. Be warned: If you are using a large panel or several at at the same time, the on board 5V connecter shouldn't be used. Wire the panel directly to a 5V supply instead.

5.4.2. Main LEDSTRIPS / Matrix wiring

The ledstrips or matrix is connected using three wires only, gnd, +5V and the data line.

5.5. Wiring tables

The wiring is not that spectacular.

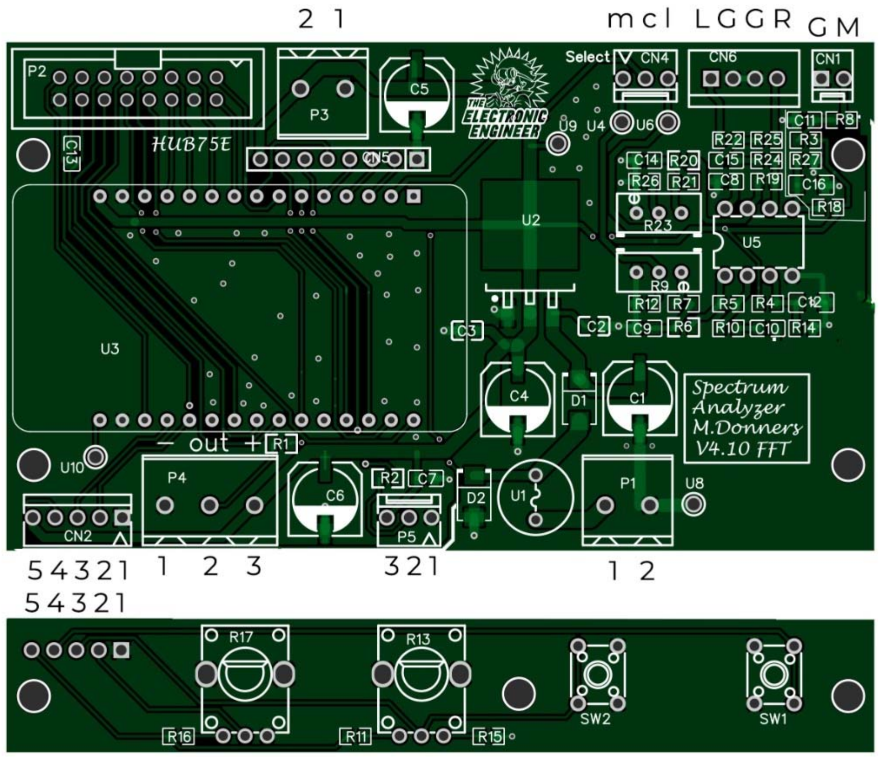
P1 Main power in	
Pin 1	+12V / 4A
Pin 2	Gnd

P3 5V power out	
Pin 1	Gnd
Pin 2	+5V
Only use for 1 panel at low brightness. Better to feed the display directly	

P5 Reserve	
Pin 1	Gnd
Pin 2	Data
Pin 3	+5V

CN2 & CN3 User interface Connect 1:1	
Pin 1	Gnd
Pin 2	dataline
Pin 3	dataline
Pin 4	dataline
Pin 5	+5V

CN6 Line In	
Pin 1	Left Channel
Pin 2	Gnd
Pin 3	Gnd
Pin 4	Right channel



P2 HUB75E interface
Use a high quality short flatcable to connect a Hub75E display

P4 Ledstrip/ Matrix pixel leds	
Pin 1	Gnd connect to gnd pin ledstrip
Pin 2	Data connect to data input ledstrip
Pin 3	+5V connect to power pin ledstrip

CN1 Microphone input	
Pin 1	Gnd
Pin 2	Microphone +

Cn4 Select switch	
Pin 1	Microphone
Pin 2	Common
Pin 3	Line in
You can use a toggle switch or make a permanent selection using a jumper	

6. Software

In general you can say that the program in running in repeated mode. Every loop, it will process data from a buffer using FFT. The buffer itself if filled with data in the background while the main program is running by using I2S direct communication. One of the perks of the ESP32. The FFT data is analysed and grouped into bars that are displayed. The ledstrip/matrix is controlled with a fastled library while the HUB75 display is driven using another great library with I2S approach. Each bar represents a frequency band and the code calculated the number of leds it should light up per bar. 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.

The unit has two switches, one to do a reset and one to change the operating mode. (Changing colours and patterns)

Although the ESP32 has a dual core and one can assign functions to a specific core, it is not used in this version because it is not needed. Although, that might change in the future when more functions are implemented. (for example a web interface)

6.1. Files

The firmware can be divided into 9 files

- FFT_ESP_V1.0 → main function and code
- FFT.h → Sepcifications and settings for the FFT functions
- I2SPlUGIN.h → Initialisation of the on board I2S audio functions
- LEDDRIVER.h → Settings that are specific for HUB75E and/or Ledstrip
- PatternsHUB75.h → All the graphic patterns specific for HUB75
- PatternsLedstrip.h → All the graphic patterns specific for Ledstrip
- Settings.h → All settings
- Fire.h → code for fire screensaver
- Logos.h → contains the logo data of The Electronic Engineer

You can download the latest sketch here:

https://github.com/donnersm/FFT_ESP32_Analyzer

6.1.1. Settings

The following settings can be changed. The name in the top of each table gives the name of the file where to find it.

LEDDRIVER.H

For Ledstrip and HUB75 settings see section 5.3

FFT.H

```
// Depending on how many bands have been defined, one of these tables will contain the frequency
// cutoffs for that "size" of a spectrum display.
// Only one of the following should be 1, rest has to be det to 0
#define bands8  0
#define bands16 0
#define bands24 1
#define bands32 0
#define bands64 0
```

FFT.H

```
// Depending on how many bands have been defined, one of these tables will contain the frequency
// cutoffs for that "size" of a spectrum display.
// Only one of the following should be 1, rest has to be det to 0
#define bands8  0
#define bands16 0
#define bands24 1
#define bands32 0
#define bands64 0
```

Settings.h

```
// Debug features should default be off!
#define PrintRAWBins        0          // set to 1 if you want to print the RAW FFT BIN values of each pass to the serial port
#define PrintADCRAW         0          // Set to 1 if you want to print the RAW ADC Values's of each pass to the serial port--> that
                                       // is a lot of samples!!
#define VisualizeAudio      0          // Sends the raw ADC values from buffer to serial plotter...that is a lot of data!!
#define CalibratieLog       0          // a tool tell help with  calibrating to noise input signal
int  DEBUG =                0 ;        // When debug=1, extra information is printed to serial port. Turn of if not needed
#define DEBUG_BUFFER_SIZE   100        // Debug buffer size

//Options Change to your likings
#define BottomRowAlwaysOn    1         // if set to 1, bottom row is always on. Setting only applies to LEDstrip not HUB75
#define Fallingspeed         5         // Falling down factor that effects the speed of falling tiles
#define LogoBoot             1         // Show logo on boot when set to 1, only works in combination with HUB75
int Peakdelay =             60;        // Delay before peak falls down to stack. Overruled by PEAKDEALY Potmeter
#define GAIN_DAMPEN          2         // Higher values cause auto gain to react more slowly
#define SecToChangePattern  10         // number of seconds that pattern changes when auto change mode is enabled
#define MAX_VU              5000       // How high our VU could max out at.  Arbitarily tuned.
int buttonPushCounter =     0;         // This number defines what pattern to start after boot (0 to 12)
bool autoChangePatterns =   false;     // After boot, the pattern will not change automatically.
int NoiseTresshold =        1500;      // this will effect the upper bands most.
#define DemoAfterSec        6000       // if no input signal during this number of milli seconds, the unit will go to demo mode
#define DemoTreshold        500        // this defines the treshold that will get the unit out of demo mode
#define BRIGHTNESSMAX       255        // Max brightness of the leds...carefull...to bright might draw to much amps!
int BRIGHTNESSMARK=         50;        // Default brightnetss, however, overruled by the Brightness potmeter
int BRIGHTNESSMIN =         20;        // Min brightness

//buttonstuf don't change unless you know what you are doing
#define ShortPress           45
#define LongPress            2000
#define LongerPress          4000
#define ButtonTimeout         200
#define ButtonSequenceRepeatTime 200


// There are also several setting for the different colors that are used in the patterns. You can change them and experiment
```

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. Make sure that your Arduino IDE is set for using the ESP32. If you don't know how to do so, google or look it a youtube video. There are some very clear instructions and setting up the IDE is not hard. You can do it! In a nutshell, it comes down to this:

1. In the Ide preferences window, look for the line: Additional Boards Manager and add the following line; https://dl.espressif.com/dl/package_esp32_index.json
2. Go to your board manager and look for ESP32 and install the ESP32 from Espressif Systems.
3. Select the correct board before you compile and you are good to go

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

Here are the libraries that you will need to install using the Arduino Library manager. These I the versions I used. Higher version might work just fine.

Adafruit_BusIO	at version 1.7.1
Adafruit_GFX_Library	at version 1.10.4
arduinoFFT	at version 1.5.6
ESP32_HUB75_LED_MATRIX_PANEL_DMA_Display	at version 2.0.5
FastLED	at version 3.4.0
FastLED_NeoMatrix	at version 1.1
Framebuffer_GFX	at version 1.0

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 and a word of thanks

No trouble so far! Will be added one at a time in time 😉

Or as Dave Plummer would say from Dave's Garage: In the meantime and in between time…catch you later next time.

Don't forget to check out his channel as he inspired me to make this build as did many others:

Dave Plummer	https://www.youtube.com/channel/UCNzszbnvQeFzObW0ghk0Ckw
Mrfaptastic	https://github.com/mrfaptastic
Scott Marley	https://www.youtube.com/user/scottmarley85
Brian Lough	https://www.youtube.com/user/witnessmenow
atomic14	https://www.youtube.com/channel/UC4Otk-uDioJN0tg6s1QO9lw

You probably got hands on this document as a result of watching a video on my youtube channel, right?

Well, if you did, I would appreciate a like, a subscribe and a comment if you like. Feedback and pat on the shoulder is what keeps me going and it motivates me to put more stuff in the public domain.

https://www.youtube.com/channel/UCm5wy-2RoXGjG2F9wpDFF3w