UNIVERSITÀ DI PISA

SCUOLA DI INGEGNERIA

5 Band Equalizer

Leonardo Portanova – Mattia Tinfena

Costruzioni Elettroniche

A.A. 2018/2019

1 INTRODUCTION

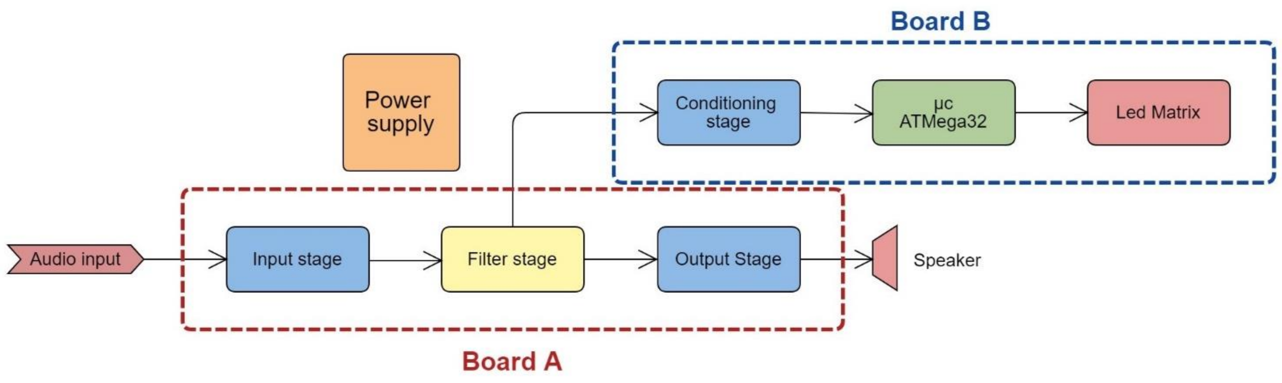
Equalization meaning is to adjust the balance between frequency components of an audio signal, this is done with an equalizer: a device that boost or reduce the various frequency ranges.
This function is particularly useful in audio production contests or in live events where there is the need to give the audio signal specific characteristics.

Normally the range of audible frequencies for human is from 20 to 20˙000 Hz so we divided that range in 5 bands.

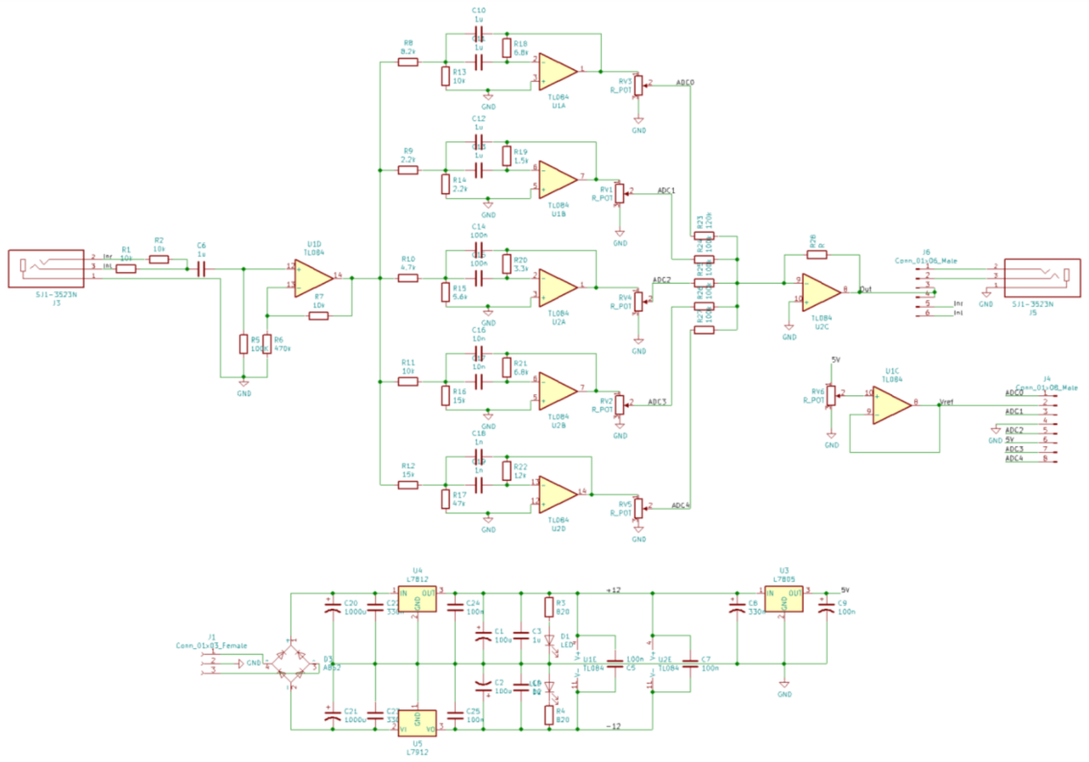
We also included a VuMeter for each band to simply visualize the effects of the equalization process in the audio signal.

2 BLOCK DIAGRAM & SCHEMATIC

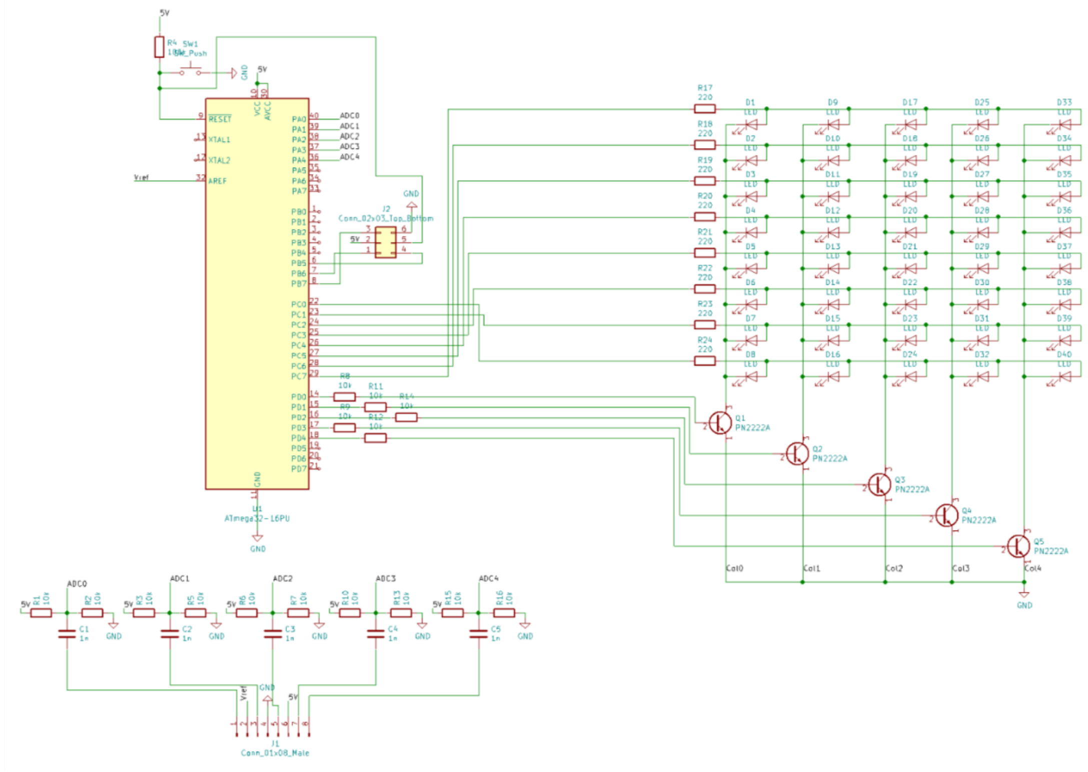
2.1 BLOCK DIAGRAM



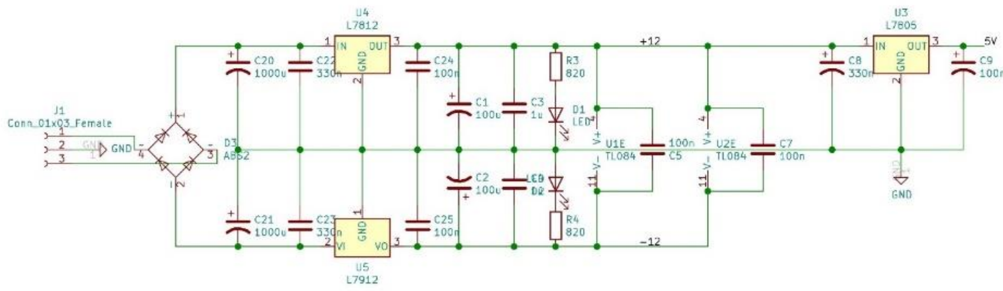
2.2 BOARD A SCHEMATIC



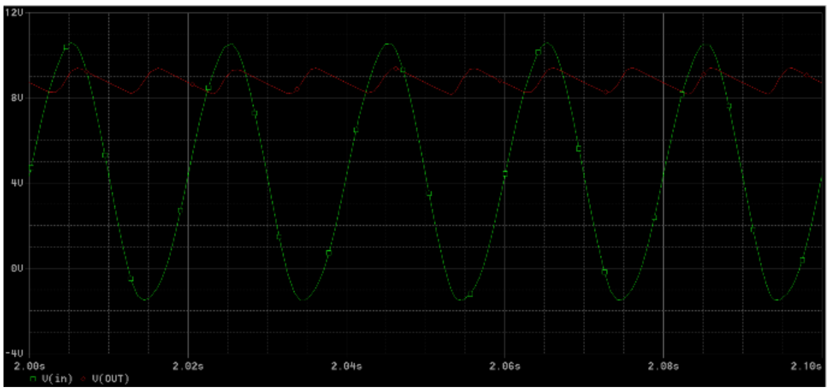
2.3 BOARD B SCHEMATIC



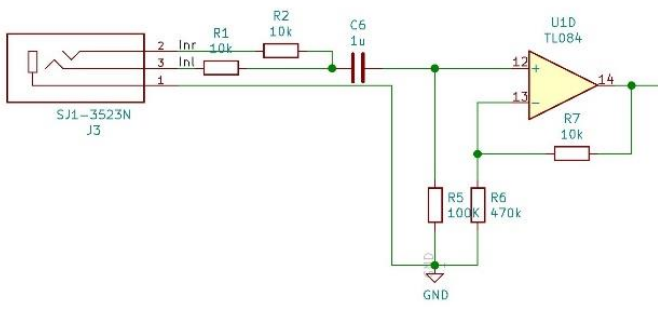
2.4 POWER SUPPLY BLOCK



Power is provided from a 24Vac transformer. To convert AC current in DC we used a full bridge rectifier and three voltage regulators to have +12Vdc and -12Vdc for the tl084 (general purpose OpAmp), and 5Vdc for the microcontroller and the led matrix. To rectify the voltage, after the full bridge rectifier, we used two 1mF capacitors dimensioned to limit the output ripple to ±1V. This value was obtained from a simulation in which we considered the current absorbed by the circuit (about 250mA).
The 100µ and 1µ capacitors (C1, C2, C3, C4) purpose is to prevent noise introduction, respectively at medium and high frequencies.

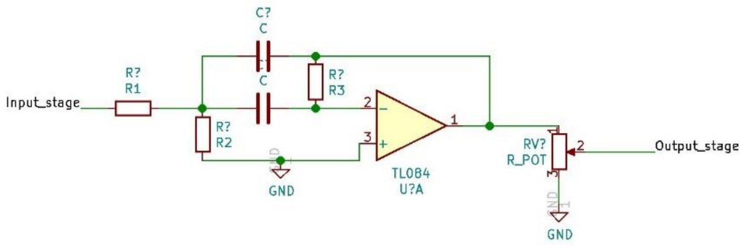


2.5 INPUT STAGE



The input stage is formed by a non-inverting buffer to uncouple the input and the filter stage. A resistor is placed in series to each channel (left and right) to sum them and prevent short-circuit which can damage the upstream circuit. A capacitor is placed before the non-inverting input of the OpAmp to remove the continuous component of signal. One resistor is placed between non-inverting terminal and GND to ensure the polarization current for the OpAmp.

2.6 FILTER STAGE



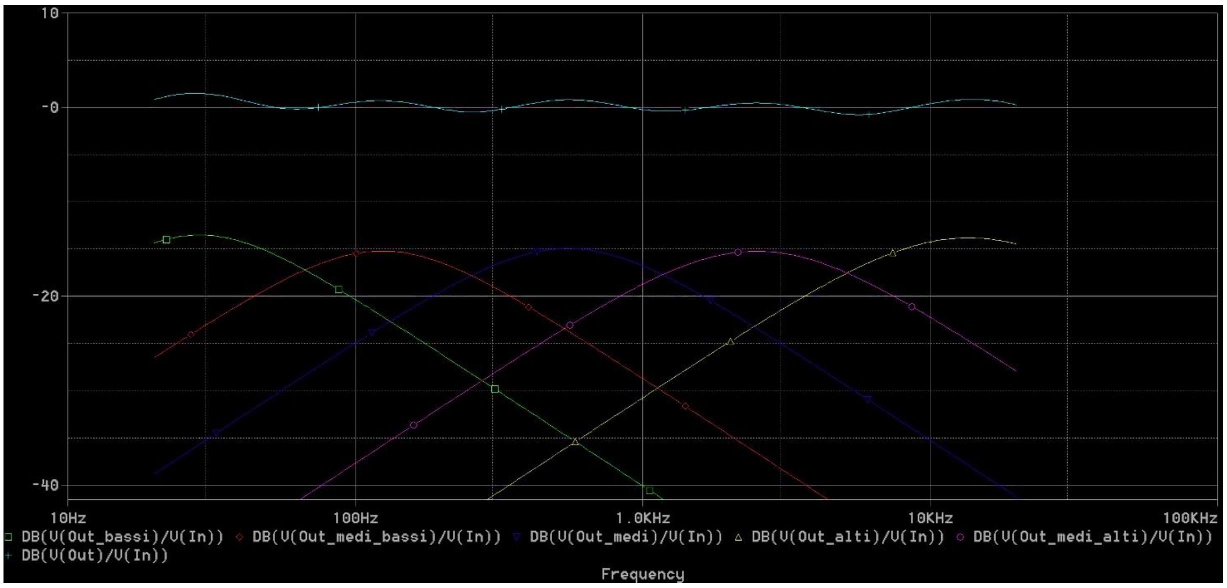
Filter stage is formed by five second order band pass filters (multiple feedback). The gain of each band can be regulated with a variable resistor which divide the output voltage of the filter. All filters have a quality factor "Q" of 0.6, and the central frequencies are: 30 Hz, 125 Hz, 500 Hz, 2.5 kHz, 14 kHz. The resistors value are chosen once the capacitors value and Q are set, with the following formulas:

$$|F_r| = \frac{Q}{R_1 C \omega_r} < 2Q^2$$

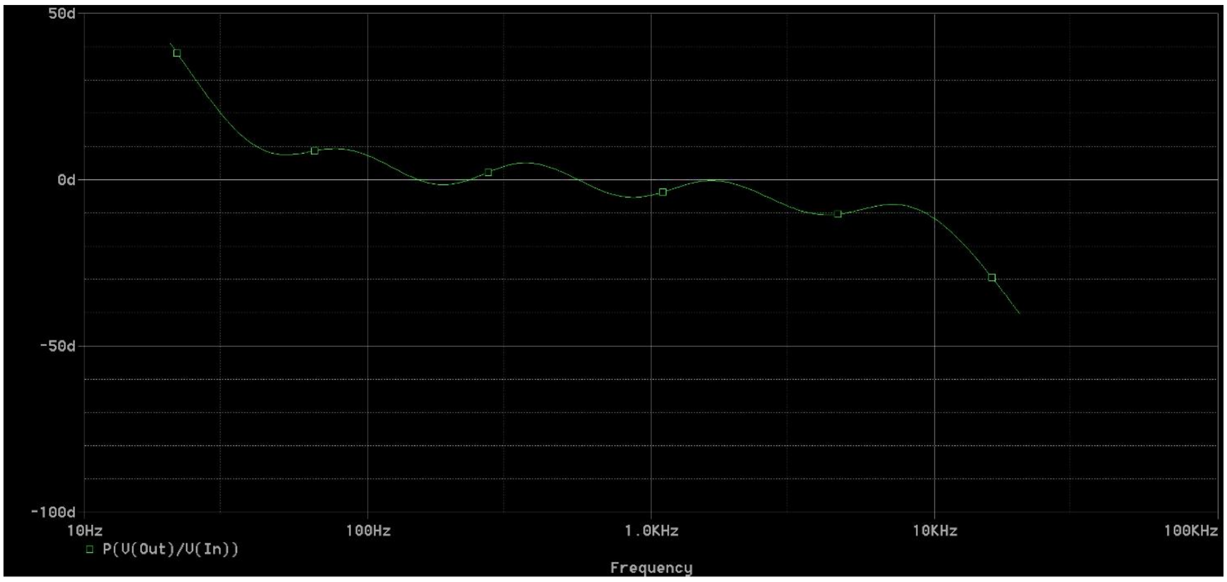
$$R1 > \frac{Q}{C\omega_r |F_r|} \quad R2 = \frac{Q}{(2Q^2 - |F_r|)\omega_r C} \quad R3 = \frac{2Q}{\omega_r C}$$

The simulation results with no reduction or boost applied are:

- Frequency response: module diagram

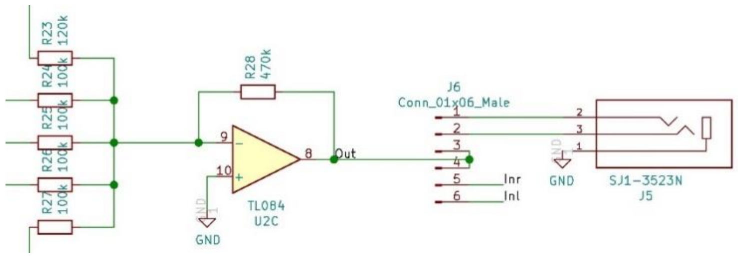


- Frequency response: phase diagram



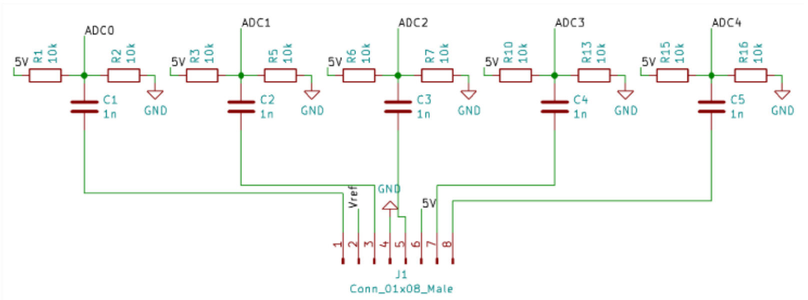
The module diagram shows an almost flat frequency response wich implies good accuracy in the signal reproduction.
The phase diagram shows that the circuit do not phase shift significally the signal, especially in the central frequencies.

2.7 Output stage



Output stage is formed by an inverting adder with a fixed gain of about 4 to compensate the attenuation of the filters. We also placed a selector to eventually exclude the equalizer.

2.8 Conditioning stage



Conditioning stage is necessary because the microprocessor's ADCs are unipolar. We used a voltage divider formed by two equal resistors to shift the average value of the signal to Vcc/2 so we have an all positive signal. We also used a capacitor in series to the signal, before the voltage divider, to uncouple the input from the ADCs and remove the DC component.
We used 10kΩ resistors and 1nF capacitors to have a pole frequency below 20Hz.

2.9 µC: ATMEGA32 STAGE

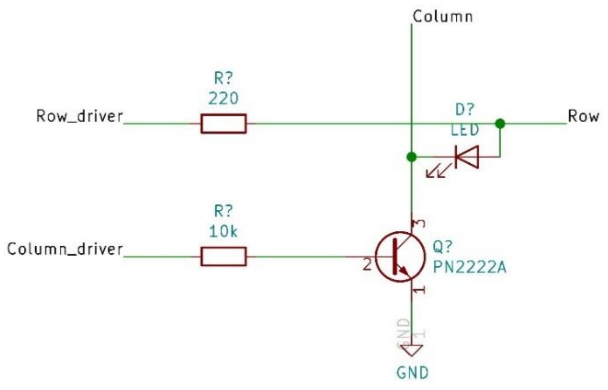
To realize the VuMeter we needed a led matrix driver. We ended up using a microcontroller to reduce the PCB's dimension and cost compared to five single dot led bar driver (one for each channel), and allows to implement some useful algorithms.

We used an AtMega32 because it has 8 channels whit 10-bit ADCs and opted for the THT version to improve repairability simplifying replacement process.

The firmware periodically controls the ADCs input and, after compensating the error caused by the average value shifting, it calculates the average value of the peaks reached by the signal and drive the led matrix with the appropriate values.

We used port-A for the ADCs, port-C to drive the led matrix's rows and port-D to drive the led matrix's columns. Port-B is purposely left unused to allow programming the microcontroller (an ISP connector is located on the board B)

2.10 LED MATRIX



To drive the led matrix, we used one npn BJT (2N2222) for each column because a single pin of the microcontroller can't handle the necessary current to drive eight LEDs at the same time (worst case: 160mA). The resistance in the path is dimensioned to limit the led current to about 20 mA.

3 PARASITIC ELEMENTS

The parasitic effects have been calculated on the longest path for each board:

$$R_{board-A} = \rho \cdot \frac{l}{s} = 0.017 \times 10^{-6} \cdot \frac{10.4 \times 10^{-2}}{0.75 \times 10^{-3} \cdot 35 \times 10^{-6}} = 67.4 m\Omega$$

$$R_{board-B} = \rho \cdot \frac{l}{s} = 0.017 \times 10^{-6} \cdot \frac{12.2 \times 10^{-2}}{0.7 \times 10^{-3} \cdot 35 \times 10^{-6}} = 84.7 m\Omega$$

The resistance values obtained are very low, as expected.

$$C_{par} = 55.6 \times 10^{-12} \cdot x \cdot \frac{1}{\ln\left(\frac{4 \cdot h}{\phi}\right)} \qquad C_{par_{board-A}} = 3.07\, pF \qquad C_{par_{board-B}} = 3.54\, pF$$

$$L_{auto} = 2 \cdot 10^{-7} \cdot x \cdot \ln\left(\frac{4 \cdot h}{\phi}\right) \qquad L_{auto_{board-A}} = 39.14\, nH \qquad L_{auto_{board-B}} = 46.76\, nH$$

$$L_{mutua} = L_{auto} \cdot \frac{1}{1+\left(\frac{S}{h}\right)^2} \qquad L_{mutua_{board-A}} = 38.9\, nH \qquad L_{mutua_{board-B}} = 46.45\, nH$$

$$\Phi_{board-A} = \sqrt{\frac{4A}{\pi}} = 182.8\ \mu m \qquad \Phi_{board-B} = \sqrt{\frac{4A}{\pi}} = 176.6\ \mu m$$

4 SIGNAL INTEGRITY

4.1 DISTRIBUTER/CONCENTRATED PARAMETERS

Since the circuit works with audio signals, the maximum frequency in the Board-A will be around 20 kHz.
The corrisponding wavelenght is:

$$\lambda = \frac{c}{f \cdot \sqrt{\varepsilon_{r_{eff}}}}$$

Where c is the speed of light and ε_r is the dielectric constant for FR-4 at the specific frequencies used.
Using the KiCAD tool:

$$\varepsilon_{r_{eff}-board-A} = 3.14$$

So we obtain: $\lambda_{board-A} = 8.46\ km$

In Board-B, the maximum frequency is set by the T_{rise} of the microcontroller's output. Since it is clocked at 1 MHz (internally) we can at least have a 1 MHz square-wave signal. Considering a 10% rise time (10 ns) we can approximate the bandwidth limit to $f_{max} = \frac{1}{2\pi T_{rise}} = 15.92\ MHz.$
Using the KiCAD tool:

$$\varepsilon_{r_{eff}-board-B} = 3.50$$

In this case the result is: $\lambda_{board-B} = 10.07\ m$

The maximum trace length until concentrate parameter can be used is:

$$\frac{\lambda_{board-A}}{6} = 1.41\ km \qquad \text{(maximum trace lenght: 10.4 cm)}$$

$$\frac{\lambda_{board-B}}{6} = 1.68m \qquad \text{(maximum trace lenght: 12.2 cm)}$$

Concentrate parameter analysis is correct and can be utilized.

4.2 RINGING ANALYSIS

The higher signal frequency on the board-A is set by the audio bandwidth limit: $f_{sig\_max\text{-}A}$ =20 kHz. On board_B we can consider $f_{sig.max\text{-}B}$=15.92 MHz, as previously seen.
If f_{sig_max} is lower than the auto-resonance frequency (f^*), ringing probelms may occour.

$$f^* = \frac{1}{2\pi\sqrt{L_{auto} \cdot C_{par}}}$$

$$f^*{}_{board-A} \approx 459\ MHz \qquad f^*{}_{board-B} \approx 391\ MHz$$

both greater than $f_{sig\_max\text{-}A}$ and $f_{sig.max\text{-}B}$.

Therefor there are not ringing problems.

5 THERMAL CONSIDERATIONS AND DESIGN

In the thermal analysis only voltage regulators (7812, 7912, 7805), operational amplifiers (TL084) and BJT (2N2222) will be considered due to their higher power dissipation relatively to other components.

- **LM7812 (package: TO-220)**
  R_{ja}= 50 °C/W
  $T_{j\text{-}max}$= 150 °C
- **LM7912 (package: TO-220)**
  R_{ja}= 50 °C/W
  $T_{j\text{-}max}$= 150 °C
- **LM7805 (package: D^2PAK)**
  R_{ja}= 62.5 °C/W
  $T_{j\text{-}max}$= 150 °C
- **TL084 (package: SOIC-14)**
  R_{ja}= 86 °C/W
  $T_{j\text{-}max}$= 125 °C
- **2N2222 (package: SOT23)**
  R_{ja}= 357 °C/W
  $T_{j\text{-}max}$= 150 °C

The worst case is set by the voltage regulators which can work with a peak of 125mA at 12V: P_d=1.5 W.

In this case: $T_j = T_a + (P_d \cdot R_{ja}) = 25°\text{C} + (1.5\ W \cdot 50\ °\text{C/W}) = 100°\text{C}$ which is significantly lower than the maximum working temperature, therefore no cooling system is required.

6 RELIABILITY

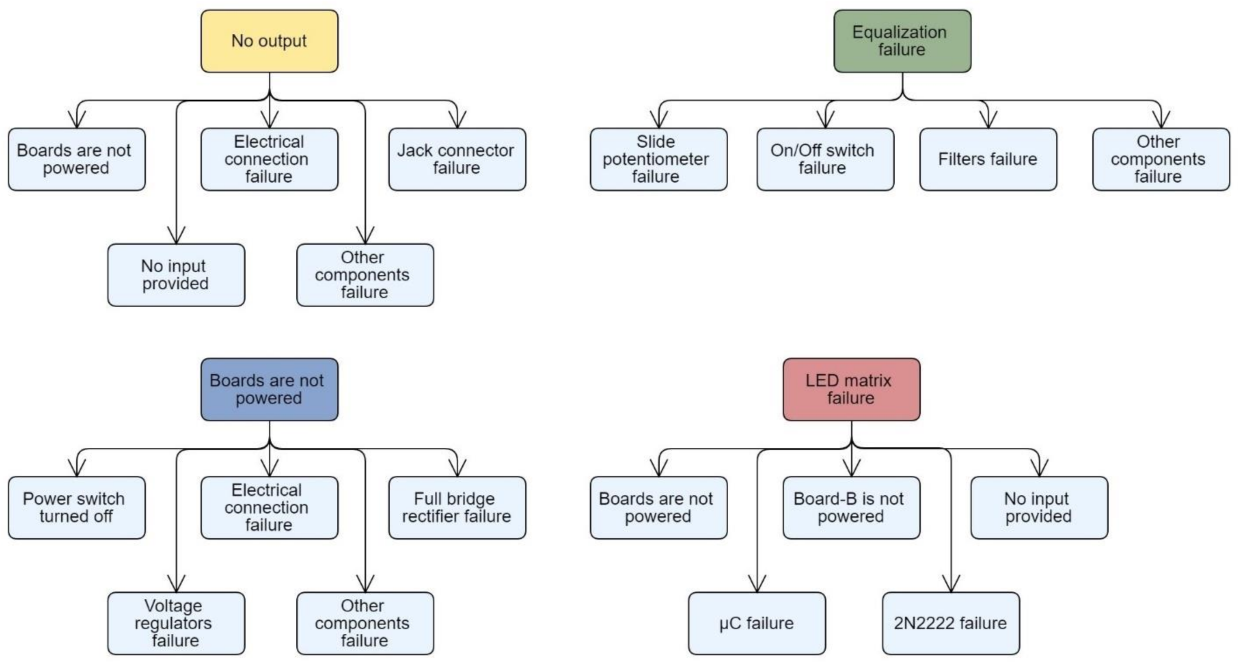
Reliability datas can be found in the component's datasheet.

- **LM7812 (package: TO-220)**
  From Texas Instruments website:
  MTBF: $4.08 \times 10^8\ hours\ (at\ 125\ °\text{C})$
- **LM7912 (package: TO-220)**
  From Texas Instruments website:
  MTBF: $4.08 \times 10^8\ hours\ (at\ 125\ °\text{C})$
- **LM7805 (package: D^2PAK)**
  From Texas Instruments website:
  MTBF: $4.08 \times 10^8\ hours\ (at\ 125\ °\text{C})$
- **TL084 (package: SOIC-14)**
  From Texas Instruments website:
  MTBF: $1.47 \times 10^9\ hours\ (at\ 125\ °\text{C})$
- **2N2222 (package: SOT23)**
  From International journal of machine learning and computing:
  MTBF: $2.5 \times 10^9\ hours\ (at\ 60\ °\text{C})$

The limit is set by the voltage regulators: $MTBF_{max} = 4.08 \times 10^8\ hours$ (46575 years) which is acceptable and redundant.

7 FAILURE MODE AND EFFECT ANALYSIS (FMEA)

7.1 FAULT TREE



7.2 FMEA

To evaluate the Risk Priority Number (RPN) the following values have been used:

- Severity Rating

1.	Low	Easily fixable or not very effective on the circuit
2.	Moderate	Persistent behaviour that may affect or damage the device
3.	High	Needs to be fixed because it may cause permanent damage

- Occurrence Rating

1.	Low	At least once in 140 hours (ten weeks with 2 hours per day)
2.	Moderate	At least once in 28 hours (two weeks with 2 hours per day)
3.	High	One or more times per day

- Detection Rating

1.	Low	Easily detectable
2.	Moderate	Low-cost instrumentations and operation needed
3.	High	High-cost instrumentations and professional operation needed

$$\boldsymbol{RPN} = Severity \times Occurence \times Detection$$

Board	Failure Mode	Effect	Possible Causes	Method of detection	S	O	D	*RPN*
A	No output	No output	Boards are not powered; Electrical connection failure; Jack connector failure; No input provided; Other components failure	No sound audible; VuMeters doesn't show anything (all LED powered off)	2	2	1	**4**
A-B	Boards are not powered	No output; Board B is not powered	Power switch turned off; Electrical connection failure; Full bridge rectifier failure; Voltage regulators failure; Other components failure	Power LED are off; no signal from the output	3	2	1	**6**
A	Equalization failure	No effects on the output signal	Slide potentiometer failure (not all bands can be equalized); On/Off switch failure (equalization not working for each band); Filters failure; Other components failure	Slide potentiometer seems not working; No effect visible on the VuMeters when regulating the boost/reduction	2	1	3	**6**
B	LED matrix failure	The effect of the equalization process can't be visualized	Boards are not powered; Board-B is not powered; No input provided; µC failure; 2N2222 failure; Other components failure	LED matrix doesn't show anything (all LED powered off)	2	2	1	**4**

8 PCB REALIZATION

The PCB layout is realized using KiCAD EDA (Electoinic Design Automation software).
Red traces are on the TOP layer, realized with copper; green traces are on the BOTTOM layer and have been realized with jump wires due to the 1 layer process.

Surface mount component were placed on the TOP layer; through hole components have been mirrored to ensure the correct working when placed in the BOTTOM layer.

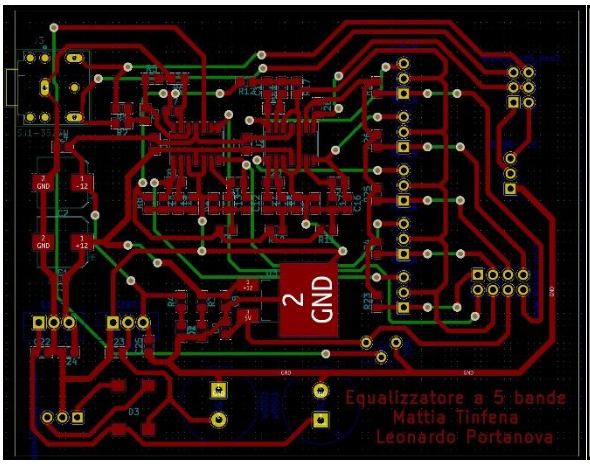


Figure 1 - PCB Diagram: Board A

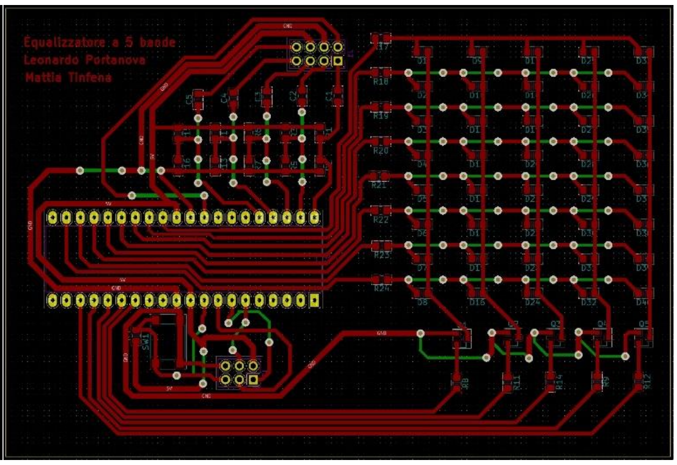


Figure 2 - PCB Diagram: Board B

The realization process started printing the mirrored TOP layer on a normal sheet of paper with a laser printer, because of the toner properties, and then reinforcing it with a "blue sheet" (Figure 3). Next we polished the copper layer to guarantee perfect sticking of the toner (Figure 4) and applied the printed sheet (Figure 5). Then we heated the board allowing the toner to transfer from the printed sheet onto the copper layer of the board (Figure 6) and dipped it in water to divide the board from the sheet (Figure 7).The result is shown below (Figure 8).

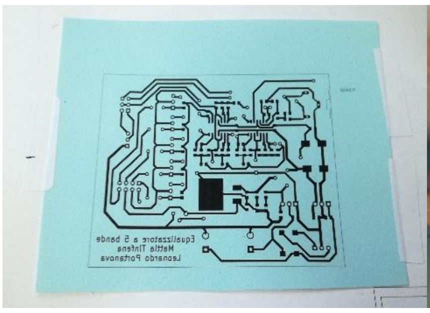


Figure 3

Figure 4

Figure 5

Figure 6

Figure 7

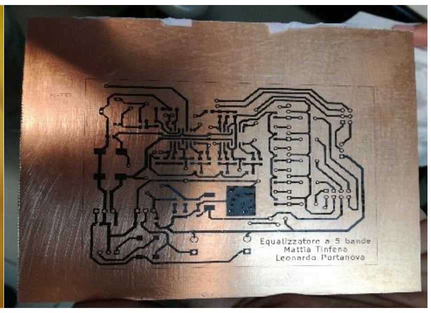


Figure 8

Next we applied the “green sheet” to the board (Figure 9); applying heat to transfer it onto the toner (and not copper) we reinforced the printing for the etching process (Figure 10). The etching process was done with a bath of caustic soda and hydrogen peroxide until all the copper was dissolved, exept the traces and pads (Figure 11). The final result is shown below (Figure 12 and 13).

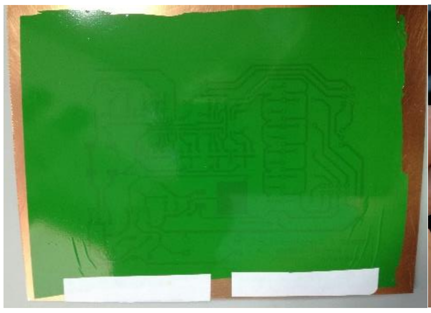


Figure 9

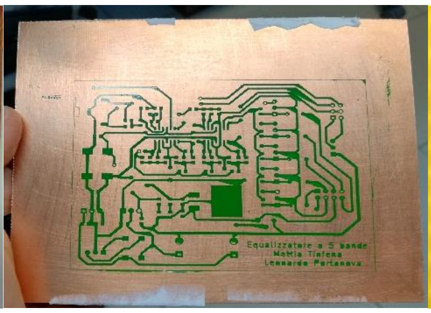


Figure 10

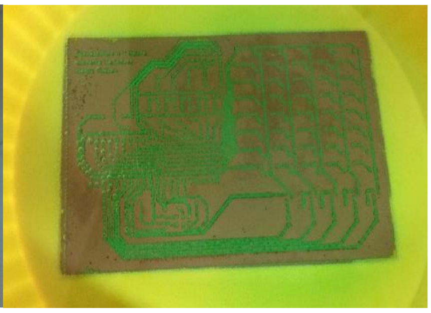


Figure 11

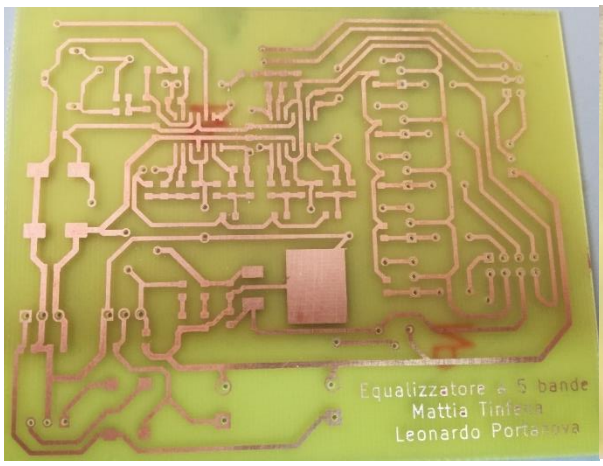


Figure 12 - Final result: Board A

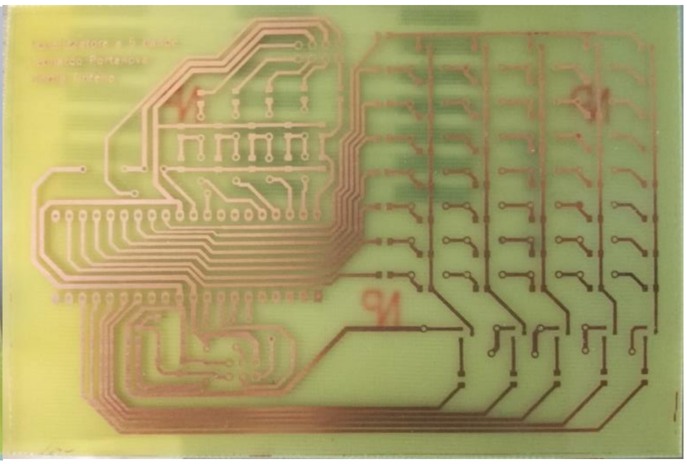


Figure 123 - Final result: Board B

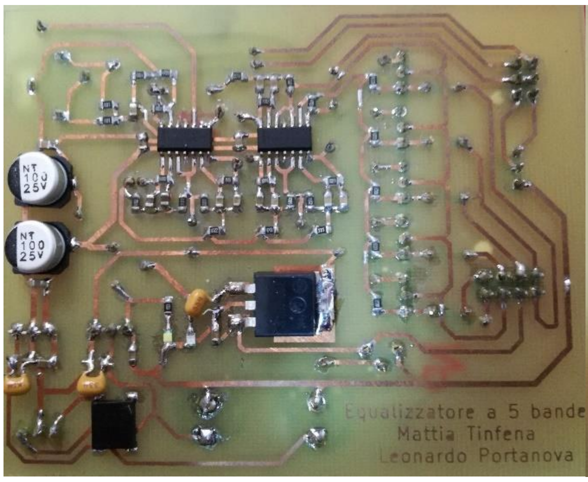


Figura 14 - Board B after 2 weeks

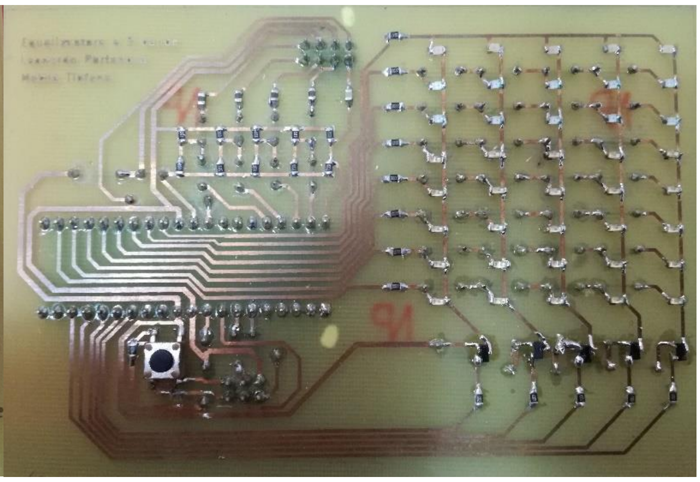


Figura 15 - Board A after 2 weeks

Comparing the boards within two weeks it can be easily noticed an evident oxidation of the copper due to the exposition to oxigen (Figure 12, 13 with Figure 14, 15). This effect can be prevent in production phase by applying a solder mask which let be exposed only soldering pads and holes.

9 DOCUMENTS AND REFERENCES

9.1 VUMETERS SOFTWARE

```
#include <avr/io.h>
#include <avr/delay.h>

#define F_CPU 1000000UL //1 MHz clock
const int rit= 100;
double valore [5]={127,127,127,127,127};
double vmedio [5]={255,255,255,255,255};


void ADC_init()
{
	ADMUX=(1<<ADLAR);		// left alignment, external Vref
	ADCSRA=(1<<ADEN)|(1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0);	// ADC enabling, Prescaler: 64
}



unsigned int ADC_read(unsigned char ch)
{
	ch= ch & 0b00000111; // channel must be b/w 0 to 7
	ADMUX =(1<<ADLAR)|ch; // selecting channel

	ADCSRA|=(1<<ADSC); // start conversion
	while(!(ADCSRA & (1<<ADIF))); // waiting for ADIF, conversion complete
	ADCSRA|=(1<<ADIF); // clearing of ADIF, it is done by writing 1 to it
	return (ADCH);
}

void port_init()
{
	DDRA=0x00;//Input port for ADCs
	DDRC=0xFF;//Output port: rows
	DDRD=0xFF;//Output port: columns
}

int pot (int esp)
{
	int ris = 1;
	for (int i = 0; i < esp; i++)
	{
		ris*=2;
	}
	return ris;
}

void intro () // initial demo
{
	PORTD = 0xff;
	PORTC = 0X00;
	int acc=0;

	for( int i = 0; i<8; i++)
	{
		int  a= pot(i);
		acc += a;
		PORTC =acc;
		_delay_ms(rit);
```

```
        }

        for (int i = 8; i>0; i--)
        {
                int  a= pot(i);
                acc -= a;
                PORTC =acc;
                _delay_ms(rit);
        }
        PORTC=0x00;
        _delay_ms(rit);

        //blink
        for (int i=0;i<2;i++)
        {
                PORTD=0xff;
                PORTC=0xff;
                _delay_ms(500);
                PORTD=0x00;
                PORTC=0x00;
                _delay_ms(500);
        }
}


void accendi (int col, int val)
{
        //row power on
        int c = pot(col);
        PORTD = c;
        //column power on
        int b=0;

        for( int i = 0; i<val; i++)
        {
                int  a= pot(i);
                b += a;
        }

        PORTC=b;
}


void elab (int col, double val) //process data from ADCs
{
        val-=vmedio[col];
        if (val<0) val*=-1;
        if(col==4)val*=110;
        else val*=65;
        if(col==1)
                {
                valore[col] = 253*valore[col]+3*val;
                valore[col] /= 256;
                }
        else
                {
                valore[col] = 230*valore[col]+26*val;
                valore[col] /= 256;
                }
        int acc = valore[col]/32;
        accendi(col,acc);
}

void valmedio(int col, double val)
```

```
{
	vmedio[col]=230*vmedio[col]+26*val;
	vmedio[col] /=256;
}

int main(void)
{
	port_init();
	ADC_init();
	intro();
	while (1)
	{
		for (int i=0; i<5;i++)
		{
			double lettura = ADC_read(i);
			valmedio(i,lettura);
			elab(i,lettura);
		}
		_delay_ms(3);
	}
}
```

9.2 References

- Texas Instruments: “Quality and Reliability” tool, available at: https://www.ti.com/quality/docs/estimator.tsp
- ATmega32 datasheet, available at: http://ww1.microchip.com/downloads/en/DeviceDoc/doc2503.pdf