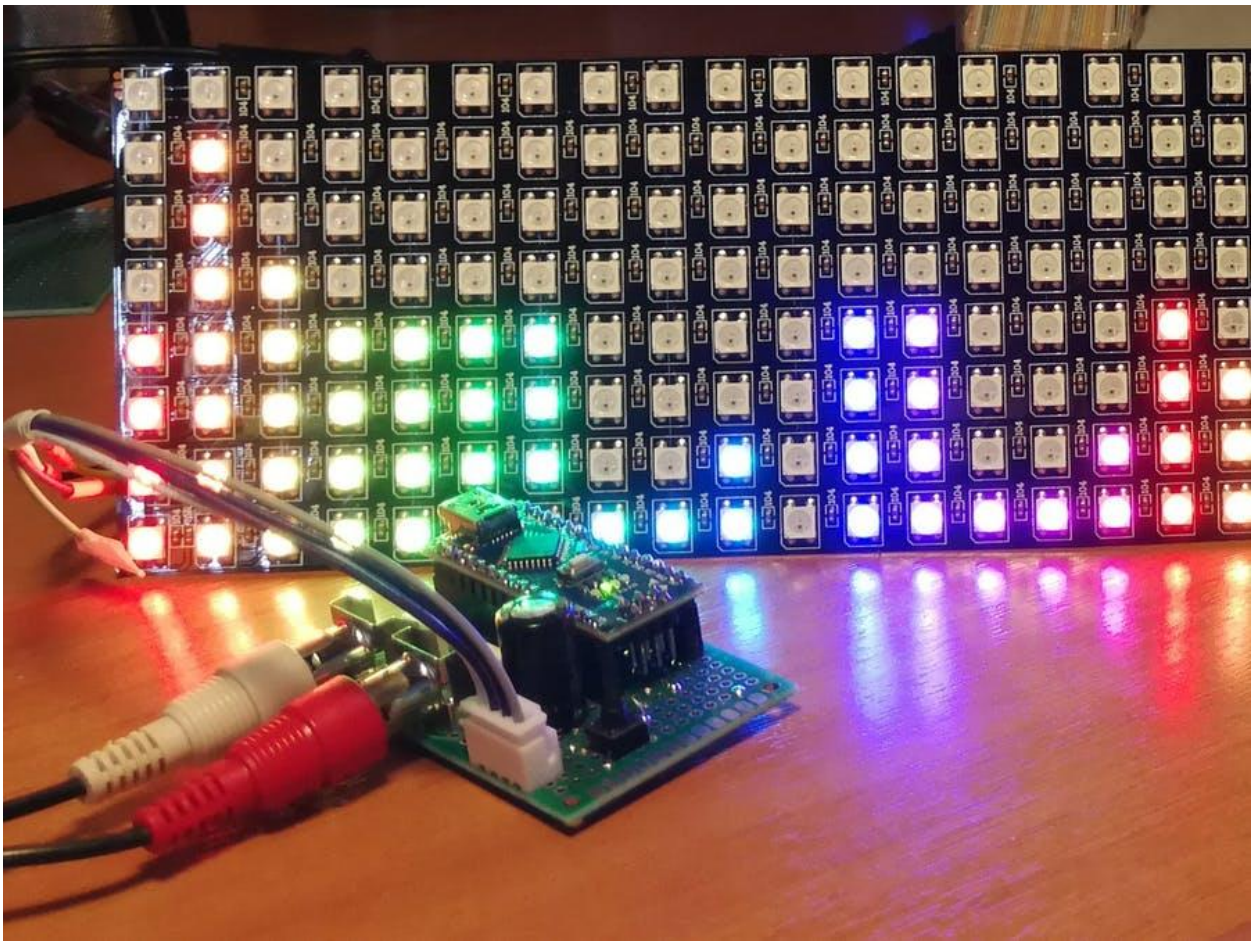


RGB 32-Band Audio Spectrum Visualizer © GPL3+

This project is nothing more than an adaptation to the WS2812B led matrix of the original project based on MAX72xx published by Shajeeb.

About this project

The Project

This project is for making a RGB 32-band audio (music) frequency spectrum visualizer using Arduino Nano and a 8x32 WS2812B RGB Led Matrix.

The original Project that inspired this

A great thanks goes to **Shajeeb** author of the original project based on the MAX72xx led matrix. I only modified the pilot part of the led matrix to adapt it to the RGB WS2812B Led Matrix.

Link to original project: 32-Band Audio Spectrum Visualizer Analyzer

The WS2812B RGB Led Matrix

Using an RGB LED matrix based on 5050 SMD high brightness LEDs, it is necessary to use an external power supply because the RGB matrix can absorb more than 10mA per LED, therefore with all the LEDs lit at maximum brightness could absorb more than 2.5 Amperes.

For this reason I have inserted a diode in series at +5V in order to be able to power Arduino in stand alone mode, when the USB cable is not connected, and to avoid Arduino being the power source of the RGB matrix, so you avoid overloading the internal circuits of the board with a current it could not supply.

To the original Project, in addition to the input diode, to protect the LED matrix input from possible voltage peaks, I also added a 390 ohm resistor in series between the Arduino pind D6 and the Data input, and a 1000 µF 12V capacitor to improve the Arduino supply voltage stability.

Hardware assembly

As visible in the main photo I made the first prototype on a 4x6 cm multi-hole board using two RCA audio sockets (soldered directly on the board) which can also be replaced with a 3.5 mm female Jack socket. The important thing to avoid hum is to make the connections between the source and the card audio input with a shielded cable. Another tip is to keep the connection between the Arduino and the led matrix as short as possible.

The code

In the end, all the software is based on the great work done by the author of the sampling procedure through the FFT library and the definitive realization of Shajeeb.

I have added two functions:

The first is **GetLedFromMatrix(...)** to map the matrix into rows and columns and to be able to address each of the 256 LEDs via row and column coordinates.

The second is the one - which I arbitrarily called **SetColumn(...)** - which turns on the LEDs of each column based on the peak value obtained by audio digitization (*values between 0 and 7*) and on the basis of the preset colors in a two-dimensional array. You can have fun changing the values and therefore the colors as you prefer. To simplify the code I used a sobroutine called Wheel() (taken from a demo attached to Adafruit's Neopixel library) that starting from a value between 0 and 255 returns an unsigned 32-bit long value to be passed directly to the setPixelColor function. On this you can play at will, keeping in mind the memory limitations of Arduino avoiding where possible the use of 32-bit variables to store RGB colors values.

Audio equalization

Furthermore, since I have run the tests with the audio coming from the sound card integrated in the PC motherboard, in order to improve the frequency response, I added a byte array of 32 values which in effect constitute an equalization curve to attenuate the bass and enhance the treble. If you don't need it, simply set the EQ_ON variable to *false* or change the attenuation level by changing the 32 values of the **eq[32]** array, a value of 100 leaves the amplitude unchanged, one less than 100 attenuates and one greater than 100 accentuates the frequency band.

Led brightness

The brightness of the matrix is preset in the code at 32 (BRIGHTNESS const). The maximum brightness value of the WS2812B matrix (on paper) is 255 but already with values greater than 100, the LED light unfortunately turns from white to pale yellow, it is probably necessary to supply the matrix through the two central red and black wires instead of at right side connector.

I'm still trying ...

Finally, if you use a maximum brightness of 64, a 1A power supply is probably enough, otherwise 2A is essential.

Future Udate

I'm working on a new version that uses the **OpenMusicLabs FHT** library which turns out to be many times faster than the Arduino FFT.

Stay tuned. :)

Please forgive my bad English, I used the google translator.

Code

```
/*
  Copyright (c) 2019 Shajeeb TM

  Permission is hereby granted, free of charge, to any person obtaining a
copy
  of this software and associated documentation files (the "Software"), to
deal
  in the Software without restriction, including without limitation the
rights
  to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
  copies of the Software, and to permit persons to whom the Software is
  furnished to do so, subject to the following conditions:
  The above copyright notice and this permission notice shall be included in
all
  copies or substantial portions of the Software.
  THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
  IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
  FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
  AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
  LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
  OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE
  SOFTWARE.

  WS2812B Led Matrix vesion by Janux
*/

#include <arduinoFFT.h>
#include <SPI.h>
#include <Adafruit_NeoPixel.h>

#define SAMPLES 64        //Must be a power of 2
#define  xres 32          // Total number of  columns in the display, must be
<= SAMPLES/2
#define  yres 8           // Total number of  rows in the display
#define ledPIN 6          // pint to control Led Matrix
#define NUM_LEDS (xres * yres)
#define BRIGHTNESS 32
#define buttonPin 5       // the number of the pushbutton pin to change
displaycolor

byte yvalue;
byte displaycolumn, displayvalue;
int peaks[xres];
byte state = HIGH;           // the current reading from the input pin
byte previousState = LOW;    // the previous reading from the input pin
byte displaycolor = 0;

//Arrays for samplig
double vReal[SAMPLES];
double vImag[SAMPLES];
byte data_avgs[xres];
arduinoFFT FFT = arduinoFFT(); // FFT object
```

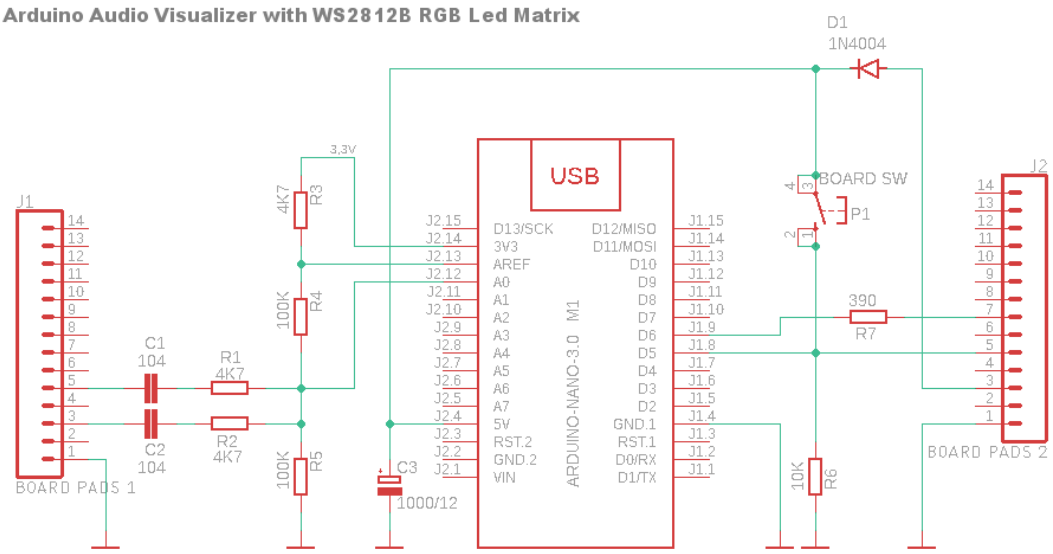
```
unsigned long lastDebounceTime = 0;   // the last time the output pin was
toggled
unsigned long debounceDelay = 100;    // the debounce time; increase if the
output flickers

// Parameter 1 = number of leds in matrix
// Parameter 2 = pin number (most are valid)
// Parameter 3 = pixel type flags, add together as needed:
//   NEO_KHZ800  800 KHz bitstream (most NeoPixel products w/WS2812 LEDs)
//   NEO_KHZ400  400 KHz (classic 'v1' (not v2) FLORA pixels, WS2811 drivers)
//   NEO_GRB     Pixels are wired for GRB bitstream (most NeoPixel products)
//   NEO_RGB     Pixels are wired for RGB bitstream (v1 FLORA pixels, not v2)
Adafruit_NeoPixel pixel = Adafruit_NeoPixel(NUM_LEDS, ledPIN, NEO_GRB +
NEO_KHZ800);

// EQ filter to attenuates bass and improves treble
// Useful on PC sound card which usually has many bass and poor high
frequency
bool EQ_ON = true; // set to false to disable eq
byte eq[32] = {50, 55, 60, 70, 75, 80, 85, 95,
               100, 100, 100, 100, 100, 100, 100, 100,
               100, 100, 100, 100, 100, 100, 100, 100,
               115, 125, 140, 160, 185, 200, 225, 255
              };

//Define color for single led, used in setColumn function, 0 for custom color
//Color are calculated by Wheel function, see below
byte colors[][8] = {
  {170, 160, 150, 140, 130, 120, 1, 1},
  {1, 5, 10, 15, 20, 25, 90, 90},
  {90, 85, 80, 75, 70, 65, 1, 1},
  {90, 90, 90, 30, 30, 30, 1, 1},
  {170, 160, 150, 140, 130, 120, 0, 0},
  {170, 160, 150, 140, 130, 120, 1, 1},
  {170, 160, 150, 140, 130, 120, 1, 1}
};

void setup() {

  pixel.begin();
  pixel.setBrightness(BRIGHTNESS);

  // Begin FFT operations
  ADCSRA = 0b11100101;      // set ADC to free running mode and set pre-
scalar to 32 (0xe5)
  ADMUX = 0b00000000;       // use pin A0 and external voltage reference
}

void loop() {

  // ++ Sampling
  for (int i = 0; i < SAMPLES; i++) {
    while (!(ADCSRA & 0x10));       // wait for ADC to complete current
conversion ie ADIF bit set
    ADCSRA = 0b11110101 ;           // clear ADIF bit so that ADC can do next
operation (0xf5)
```

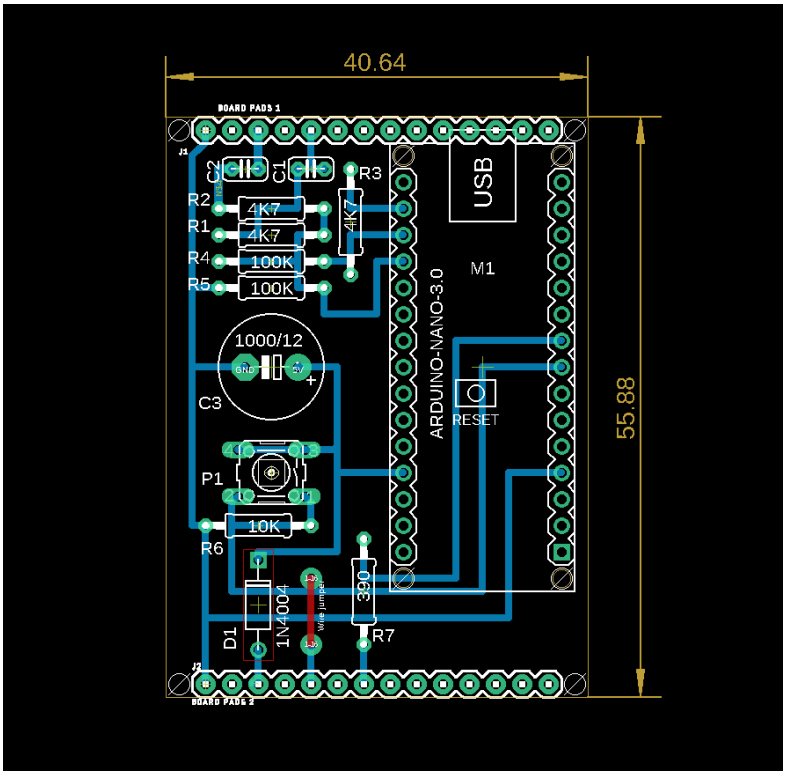
```
    int value = ADC - 512 ;         // Read from ADC and subtract DC offset
caused value
    vReal[i] = value / 8;           // Copy to bins after compressing
    vImag[i] = 0;
  }

  FFT.Windowing(vReal, SAMPLES, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
  FFT.Compute(vReal, vImag, SAMPLES, FFT_FORWARD);
  FFT.ComplexToMagnitude(vReal, vImag, SAMPLES);

  // -- FFT
  // ++ re-arrange FFT result to match with no. of columns on display (xres)
  int step = (SAMPLES / 2) / xres;
  int c = 0;
  for (int i = 0; i < (SAMPLES / 2); i += step) {
    data_avgs[c] = 0;
    for (int k = 0 ; k < step ; k++) {
      data_avgs[c] = data_avgs[c] + vReal[i + k];
    }
    data_avgs[c] = data_avgs[c] / step;
    c++;
  }

  // ++ send to display according measured value
  for (int i = 0; i <xres; i++) {
    if (EQ_ON)
      data_avgs[i] = data_avgs[i] * (float)(eq[i]) / 100; //apply eq filter
    data_avgs[i] = constrain(data_avgs[i], 0, 80);        // set max & min
values for buckets to 0-80
    data_avgs[i] = map(data_avgs[i], 0, 80, 0, yres);     // remap averaged
values to yres 0-8
    yvalue = data_avgs[i];
    peaks[i] = peaks[i] - 1;                              // decay by one
light
    if (yvalue > peaks[i]) peaks[i] = yvalue;             //save peak if >
previuos peak
    yvalue = peaks[i];
    displaycolumn = i;
    displayvalue = yvalue;
    setColumn(displaycolumn, displayvalue);               // draw buckets
  }
  pixel.show();                                           // show buckets
  displaycolorChange();                                   // check if button
pressed to change color mode
}

//----------------------------------------------------------------
// Light leds of x column according to y value
void setColumn(byte x, byte y) {
  byte led, i;

  for (i = 0; i < yres; i++) {
    led = GetLedFromMatrix(x, i); //retrieve current led by x,y coordinates
    if (peaks[x] > i) {
      switch (displaycolor) {
        case 4:
          //put zero 0 on array value to customize peaks color
```

```
          if (colors[displaycolor][i] > 0) {
            //normal color defined on color array
            pixel.setPixelColor(led, Wheel(colors[displaycolor][i]));
          }
          else {
            //custom color for peaks only with 0 on array value
            pixel.setPixelColor(led, 255, 255, 255); //Led number, R, G, B
values
          }
          break;

        case 5:
          //change color by column
          pixel.setPixelColor(led, Wheel(x * 16));
          break;

        case 6:
          //change color by row
          pixel.setPixelColor(led, Wheel(i * 36));
          break;

        default:
          //display color set -> displaycolor from 0 to 3
          //color are defined on color array
          pixel.setPixelColor(led, Wheel(colors[displaycolor][i]));
      }//END SWITCH
    }
    else {
      pixel.setPixelColor(led, 0);
    }
  }
}

//=====================================================================
// Calculate a led number by x,y coordinates
// valid for WS2812B with serpentine layout placed in horizzontal
// and zero led at bottom right (input connector on the right side)
// input value: x=0-31, y=0-7, return a led number from 0 to 255
//=====================================================================
byte GetLedFromMatrix(byte x, byte y) {
  x = xres - x - 1;
  if (x & 0x01) {
    //Odd columns increase backwards
    return ((x + 1) * yres - y - 1);
  }
  else {
    //Even columns increase normally
    return ((x + 1) * yres - yres + y);
  }
}
//=====================================================================

void displaycolorChange() {
  int reading = digitalRead(buttonPin);
  if (reading == HIGH && previousState == LOW && millis() - lastDebounceTime
> debounceDelay) // works only when pressed
  {
```

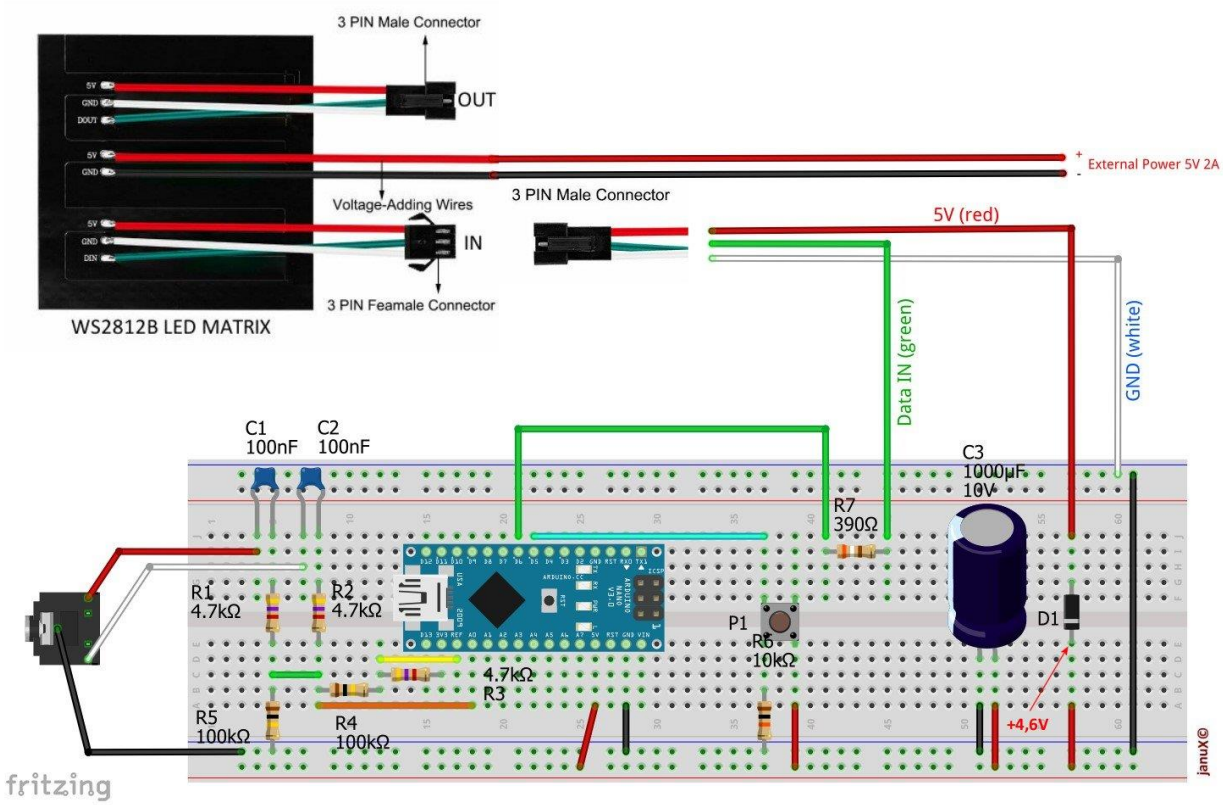
```
    displaycolor++;
    if (displaycolor > 6) displaycolor = 0;
    lastDebounceTime = millis();
  }
  previousState = reading;
}

/* Utility from Adafruit Neopixel demo sketch
   Input a value 0 to 255 to get a color value.
   The colours are a transition R - G - B - back to R.*/
unsigned long Wheel(byte WheelPos) {
  WheelPos = 255 - WheelPos;
  if (WheelPos < 85) {
    return pixel.Color(255 - WheelPos * 3, 0, WheelPos * 3);
  }
  if (WheelPos < 170) {
    WheelPos -= 85;
    return pixel.Color(0, WheelPos * 3, 255 - WheelPos * 3);
  }
  WheelPos -= 170;
  return pixel.Color(WheelPos * 3, 255 - WheelPos * 3, 0);
}
```

Custom parts and enclosures





Schematics


3 PIN Male Connector
OUT
5V
GND
DOUT
5V
GND
Voltage-Adding Wires
IN
5V
GND
DIN
3 PIN Feamale Connector
WS2812B LED MATRIX
External Power 5V 2A
3 PIN Male Connector
5V (red)
Data IN (green)
GND (white)
C1
100nF
C2
100nF
C3
1000µF
10V
R7
390Ω
R1
4.7kΩ
R2
4.7kΩ
4.7kΩ
R3
P1
R6
10kΩ
D1
R5
100kΩ
R4
100kΩ
+4,6V
fritzing
januX©