2 x 16-Band Audio Spectrum Analyzer with LCD © GPL3+

2 x 16-band audio spectrum analyzer with Arduino Nano and 2 x 16 chars LCD display. This project is based on Shajeeb's project.

About this project

This little and easy-to-do project is based on an idea to represent audio spectrum data: 32-Band Audio Spectrum Visualizer Analyzer by Shajeeb. However, I have a 2 x 40chars big LCD, and I don't wanted to create LED based bars, plus not willing to use additional hardwares. Additionally, my friends asked me to create a bit smaller version. So I changed the codes and created a 2 x 16 bars, stereo audio spectrum analyzer.

The code is changed to read data from analogue pin 0 and pin 1. Changed the hum/noise ellimination level and the voltage reference. You can change it for your needs later on as well.

https://youtu.be/kuIKAfXzQjg

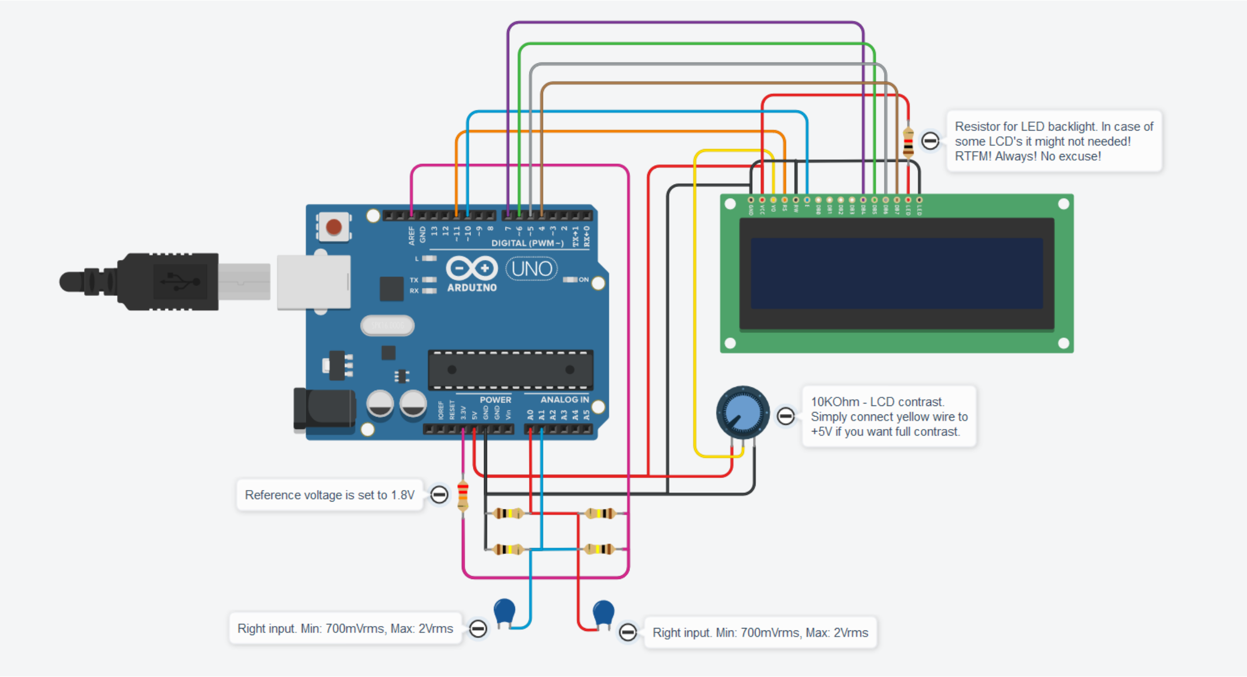
Code

```
/*
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.
*/

/*
Changed by ThomAce
*/

#include <arduinoFFT.h>
#include <LiquidCrystal.h>

#define SAMPLES 64    //Must be a power of 2

#define  xres 16      // Total number of  columns in the display, must be <=
SAMPLES/2
#define  yres 8       // Total number of  rows in the display

LiquidCrystal lcd(11, 10, 7, 6, 5, 4); // pins to LCD

//LCD Bars.
byte v1[] = {
  B00000, B00000, B00000, B00000, B00000, B00000, B00000, B11111
};
byte v2[] = {
  B00000, B00000, B00000, B00000, B00000, B00000, B00000, B11111
};
byte v3[] = {
  B00000, B00000, B00000, B00000, B00000, B11111, B11111, B11111
};
byte v4[] = {
  B00000, B00000, B00000, B00000, B11111, B11111, B11111, B11111
};
byte v5[] = {
  B00000, B00000, B00000, B11111, B11111, B11111, B11111, B11111
};
byte v6[] = {
  B00000, B00000, B11111, B11111, B11111, B11111, B11111, B11111
```

```
};
byte v7[] = {
  B00000, B11111, B11111, B11111, B11111, B11111, B11111, B11111
};
byte v8[] = {
  B11111, B11111, B11111, B11111, B11111, B11111, B11111, B11111
};
byte v9[] = {
  B00000, B00000, B00000, B00000, B00000, B00000, B00000, B00000
};

int MY_ARRAY[]={0, 1, 2, 3, 4, 5, 6, 7, 8};

double vReal[SAMPLES];
double vImag[SAMPLES];
char data_avgs[xres];

double vRReal[SAMPLES];
double vRImag[SAMPLES];
char Rdata_avgs[xres];

int yvalue;
int displaycolumn , displayvalue;
int peaks[xres];
int Rpeaks[xres];

int steps = (SAMPLES / 2) / xres;

unsigned long lastDebounceTime = 0;  // the last time the output pin was
toggled
unsigned long debounceDelay = 50;    // the debounce time; increase if the
output flickers

arduinoFFT FFT = arduinoFFT();                                    // FFT
object

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

  lcd.createChar(1, v1);
  lcd.createChar(2, v2);
  lcd.createChar(3, v3);
  lcd.createChar(4, v4);
  lcd.createChar(5, v5);
  lcd.createChar(6, v6);
  lcd.createChar(7, v7);
  lcd.createChar(8, v8);
  lcd.createChar(9, v9);
  lcd.begin(xres, 2);
  lcd.clear();

  String loading = "LOADING..    [0%]";
  int percentage = 0;
  for (int i = 0; i < xres; i++){
    lcd.setCursor(0, 0);

    percentage = (int) ((i / (float)xres) * 100);

    if (i < (xres / 3) && percentage % 2 == 0)
    {
      loading = "LOADING.  [" + String(percentage) + "%]";
      lcd.print(loading);
    }
    else if (i < ((xres / 3) * 2) && percentage % 2 == 0)
    {
      loading = "LOADING..[" + String(percentage) + "%]";
      lcd.print(loading);
    }
    else if (percentage % 2 == 0)
    {
      loading = "LOADING...[" + String(percentage) + "%]";
      lcd.print(loading);
    }

    for (int load = 0; load <= i; load++)
    {
      lcd.setCursor(load, 1);
      lcd.write(8);
    }

    delay(50);
  }

  lcd.setCursor(0, 0);
  loading = "LOADING...[100%]";
  lcd.print(loading);

  delay(500);                                     // wait to get reference
voltage stabilized and show the progress a bit longer time :)
  lcd.clear();
}


void Sampling(byte ADCBit, bool Right)
{
  ADMUX = ADCBit;//0b00000000;
  int value = 0;
  // ++ 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)

    value = ADC - 128; //- 256;// - 512 ;        // Read from ADC and
subtract DC offset caused value

    if (Right)
    {
      vRReal[i]= value / 8;                       // Copy to bins after 
compressing
      vRImag[i] = 0;
    }
    else
    {
      vReal[i]= value / 8;                        // Copy to bins after 
compressing
      vImag[i] = 0;
    }
  }
  // -- Sampling
}

void loop() {
  Sampling(0b00000001, false); //Left channel on pin 0
  Sampling(0b00000000, true); //Right channel on pin 0

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

  // ++ FFT
  FFT.Windowing(vRReal, SAMPLES, FFT_WIN_TYP_HAMMING, FFT_FORWARD);
  FFT.Compute(vRReal, vRImag, SAMPLES, FFT_FORWARD);
  FFT.ComplexToMagnitude(vRReal, vRImag, SAMPLES);
  // -- FFT

  // ++ re-arrange FFT result to match with no. of columns on display ( xres 
)
  Display(0, ReArrange(steps, data_avgs, vReal), peaks);
  Display(1, ReArrange(steps, Rdata_avgs, vRReal), Rpeaks);
}

char * ReArrange(int steps, char * dataAvgs, double * realValues)
{
  int c = 0;

  for(int i = 0; i < (SAMPLES / 2); i += steps)
  {
    dataAvgs[c] = 0;

    for (int k = 0 ; k < steps ; k++)
    {
      dataAvgs[c] = dataAvgs[c] + realValues[i + k];
    }

    dataAvgs[c] = dataAvgs[c] / steps;
    c++;
  }

  return dataAvgs;
}

void Display(int line, char * data_avgs, int * data_peaks)
```

```
{
  displaycolumn = 0;
  displayvalue = 0;
  yvalue = 0;

  // ++ send to display according measured value
  for(int i = 0; i < xres; i++)
  {
    data_avgs[i] = constrain(data_avgs[i], 0, 80);            // set max &
min values for buckets
    data_avgs[i] = map(data_avgs[i], 0, 80, 0, yres);        // remap
averaged values to yres
    yvalue = data_avgs[i];

    data_peaks[i] = data_peaks[i] - 1;    // decay by one light

    if (yvalue > data_peaks[i])
      data_peaks[i] = yvalue ;

    yvalue = data_peaks[i];

    lcd.setCursor(displaycolumn, line);
    if (MY_ARRAY[yvalue] == 0)
      lcd.write(" ");
    else
      lcd.write(MY_ARRAY[yvalue]);

    displaycolumn++;
  }
}
```

Schematics



Resistor for LED backlight. In case of some LCD's it might not needed! RTFM! Always! No excuse!

10KOhm - LCD contrast. Simply connect yellow wire to +5V if you want full contrast.

Reference voltage is set to 1.8V

Right input. Min: 700mVrms, Max: 2Vrms

Right input. Min: 700mVrms, Max: 2Vrms