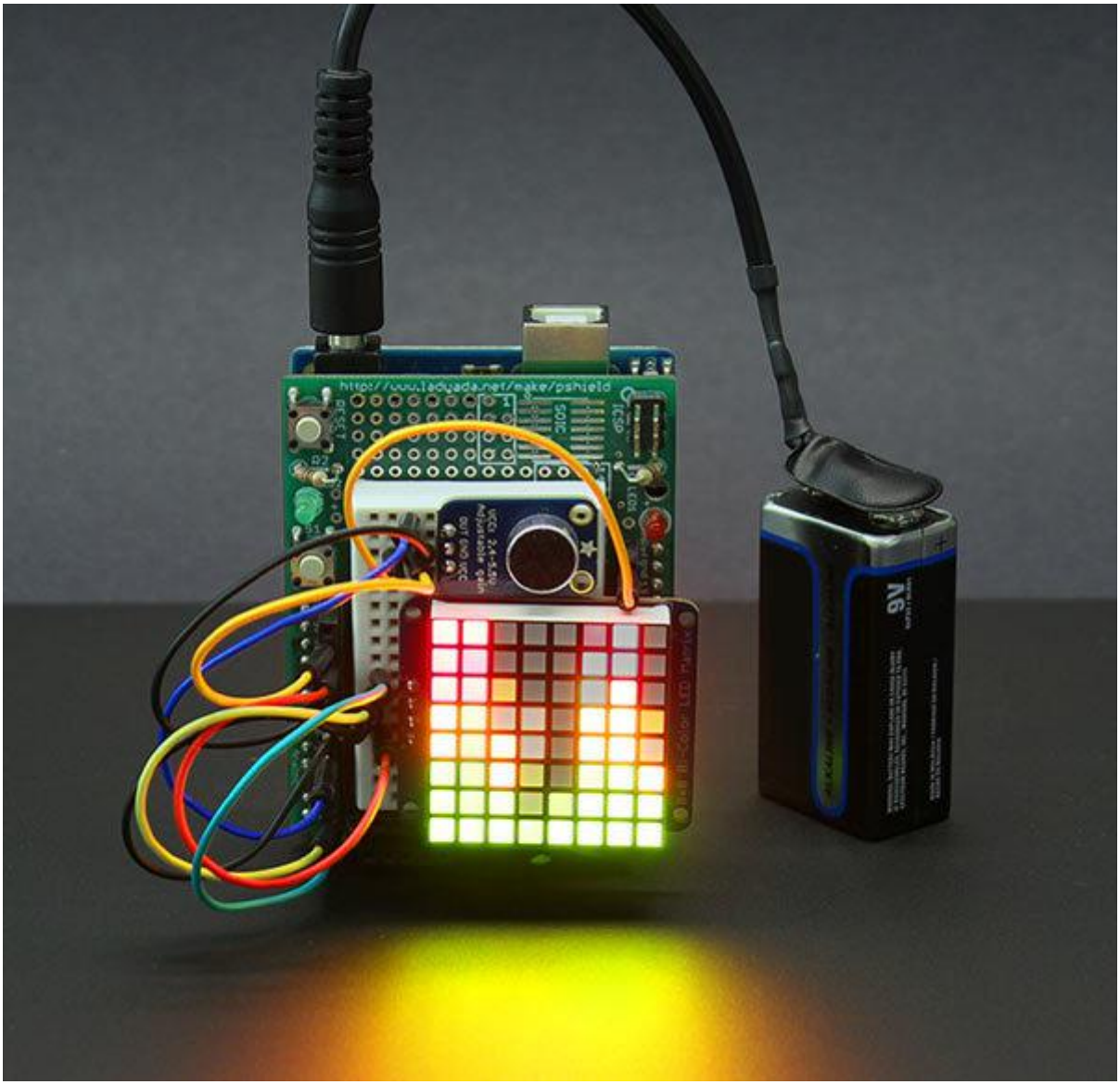
Tiny Arduino Music Visualizer

Created by Phillip Burgess



https://learn.adafruit.com/piccolo

Last updated on 2023-08-29 02:13:23 PM EDT

Table of Contents

Overview

UPDATE: this is an older guide designed for boards like the Arduino Uno and Adafruit METRO 328. It relies on assembly language code that's specific to 8-bit AVR microcontrollers and will not work with boards based on other chips (ESP32, RP2040, SAMD, etc.).

(Music: The Owl Named Orion by Dan-O at DanoSongs.com)

Here's an easy-to-build project that really packs a lot of blinkenlight for the effort: a little pocket-size music visualizer we call "Piccolo."

Set Piccolo next to the telly or some speakers and you'll see the lights respond to music and sound — lowest notes toward the left end of the graph, highest notes toward the right.

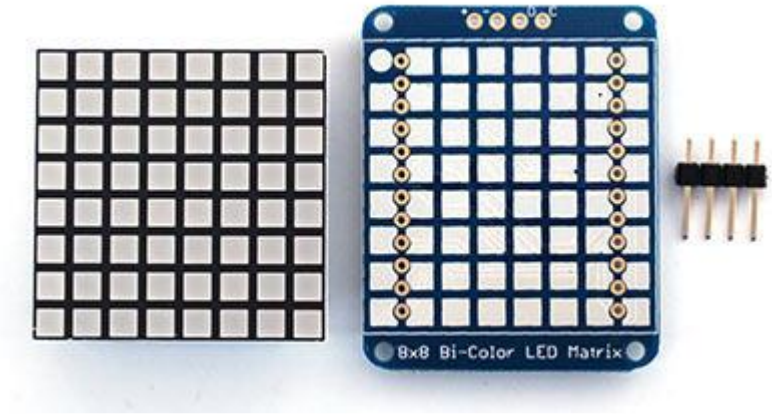
Technically this would be called a "spectrum analyzer," but as this is not a precision scientific instrument, we're more comfortable labeling it a "visualizer." It's strictly for show.

This intermediate Arduino project shows a clear progression from input to processing and then output in a package that's appealing and easy for minds to grasp: music and lights. It's not abstract or "science-y" unless you choose to peel back the layers...

Wiring

Begin by assembling the LED matrix and "backpack" board as described in the Adafruit LED Backpacks tutorial ().

If using the LED matrix for the first time, we very strongly recommend working through that full tutorial first. This will let you test and confirm that the Matrix is properly assembled before moving on to this more advanced project.

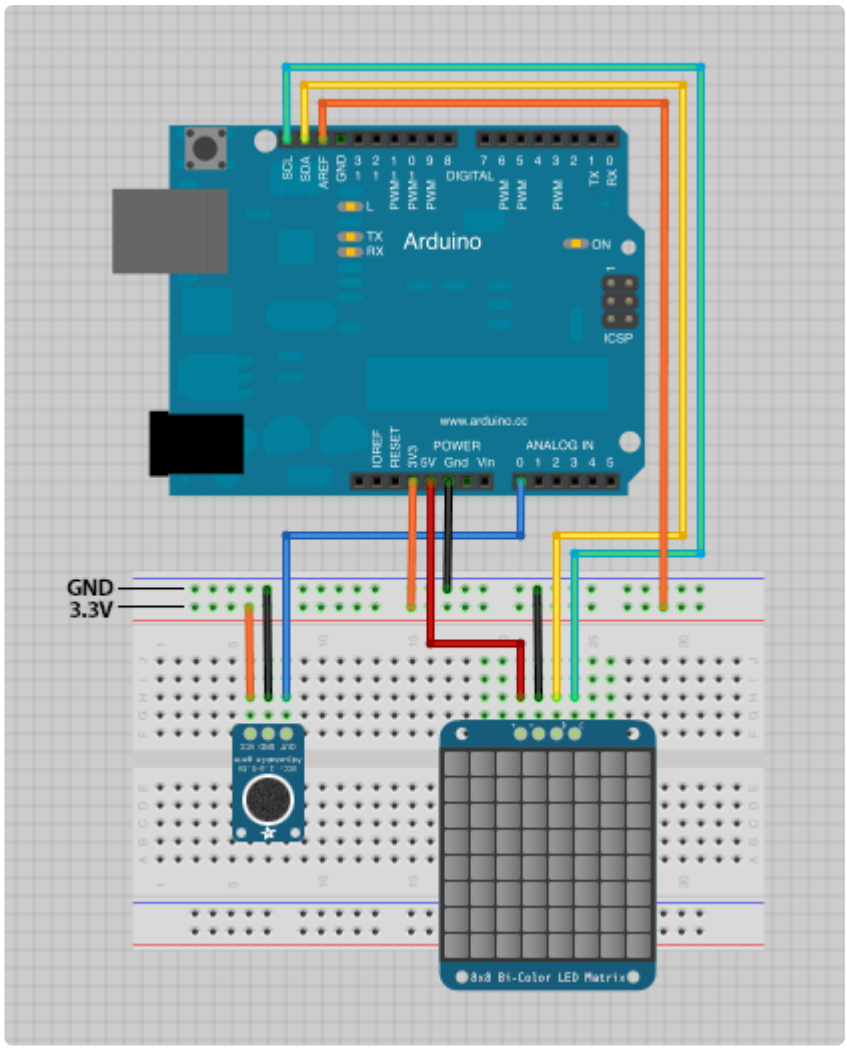


If working with a breadboard as shown below, solder row headers on to the microphone breakout and LED matrix backpack boards (three pins on the former, four on the latter). You can optionally use 90 degree headers if you'd like either component to be positioned standing up.

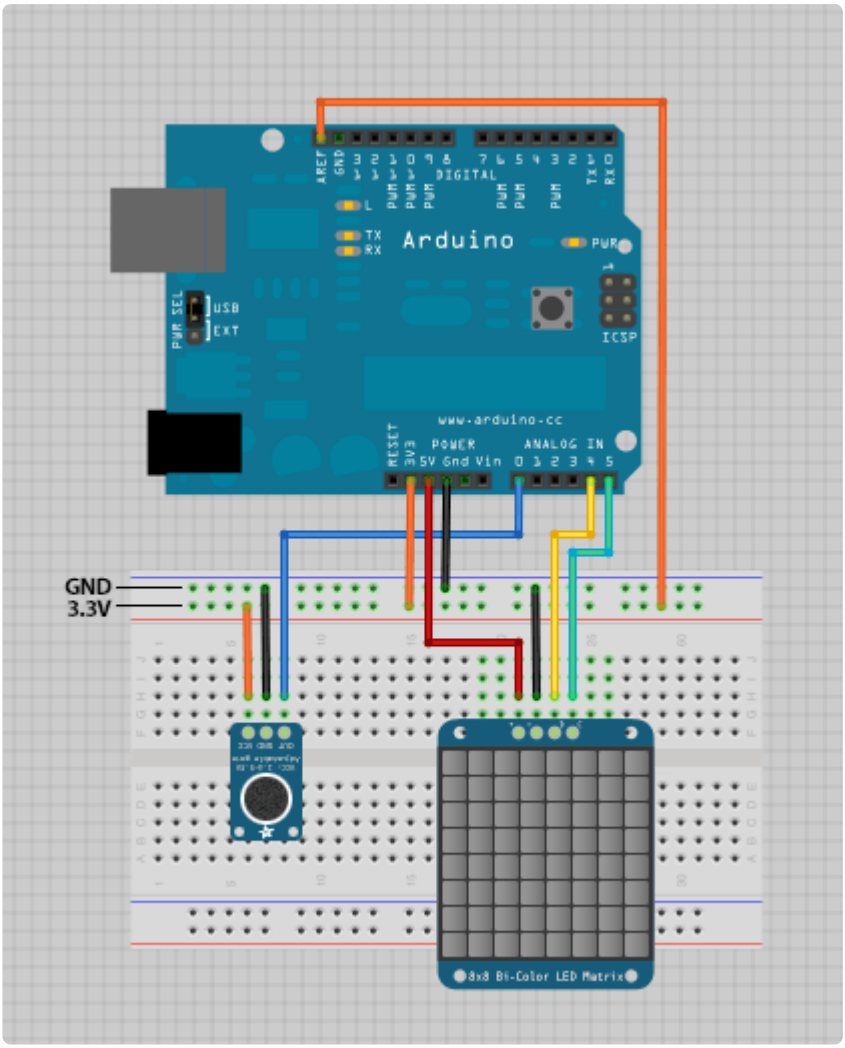
Thanks to the "smarts" of the LED matrix backpack, this whole project requires fewer than a dozen connections total:

- Connect the Arduino's 3.3V pin both to the mic amp VCC pin and the Arduino's AREF pin. We use one of the power rails on the breadboard to route 3.3 Volts to both locations. The AREF connection is very important — don't overlook this!
- Connect the Arduino 5V pin to the LED matrix + pin.
- Connect Arduino GND pin to both the mic amp GND pin and the LED matrix – pin. You can use a breadboard power rail, or the Arduino has multiple GND pins available.
- Connect Arduino analog pin 0 to the mic amp OUT pin.
- Connect Arduino pins SDA and SCL to the matrix backpack D (data) and C (clock) pins, respectively. Earlier Arduino boards don't include SDA and SCL pins — instead, use analog pins 4 (for SDA) and 5 (SCL).

For a current "R3" Arduino board (such as the Arduino Leonardo or more recent Arduino Uno and Mega boards), the wiring should resemble this:



If using a “classic” Arduino board — an Arduino Uno version R1 or R2, Arduino Duemilanove or Diecimila (with 328P chip), the matrix data and clock lines should be connected to analog pins 4 and 5:



Note: the project only works with 8-bit AVR boards. It relies on a library that's written in assembly language, and won't compile on newer 32-bit boards like Arduino Zero or Adafruit Metro Express.

Piece of cake!

You can power the Arduino from the USB connection or using a battery or power supply connected to the DC jack. The LED matrix draws a couple hundred milliamps at most, so we can safely power it through the Arduino.

Code

This button will download a folder containing all the code needed for this project:

Download "Piccolo" Arduino source code

The code can also be found on Github ().

After uncompressing the ZIP file, inside you'll find a folder called Piccolo. This can be moved to your Arduino documents folder if you like. The Piccolo.ino file in there is the main project source code — open this in the Arduino IDE.

Two libraries are required for this: Adafruit_LEDBackpack and Adafruit_GFX. If you worked through the basic LED matrix backpack guide first, these should already be present. Otherwise, both can be found in the Arduino Library Manager (Sketch→Include Library→Manage Libraries...)

With the Piccolo sketch open, select your Arduino board type and serial port from the Tools menu. Then click the Upload button. After a moment, if all goes well, you'll see the message "Done uploading." The project should now be responsive to sound...try clapping your hands!

Troubleshooting

If there's no response from the device, try the following:

- Test the matrix using the example code from the Adafruit_LEDBackpack Library. If there's no response, the clock and data pins might be swapped, or the matrix may have been soldered to the board backwards.
- Double-check all wiring against the diagrams. Did you include the 3.3V-to-AREF connection? D and C pins from the matrix backpack should connect to SDA and SCL on newer Arduinos, or analog pins 4 and 5 on older boards.
- The gain on the mic amplifier may be set low. There's a dial on the back of the board that can be adjusted with a small screwdriver.
- Try increasing the music volume.

Principle of Operation

Using the normal Arduino analogRead() function would be much too slow for sampling audio. Instead, a feature of the microcontroller's analog-to-digital converter called free-run mode is utilized. This automatically takes repeated analog samples at precise intervals...about 9.6 KHz for this project, the maximum a 16 MHz Arduino can handle with 10-bit samples.

This is strictly mono. There isn't enough RAM or processor oomph for stereo.

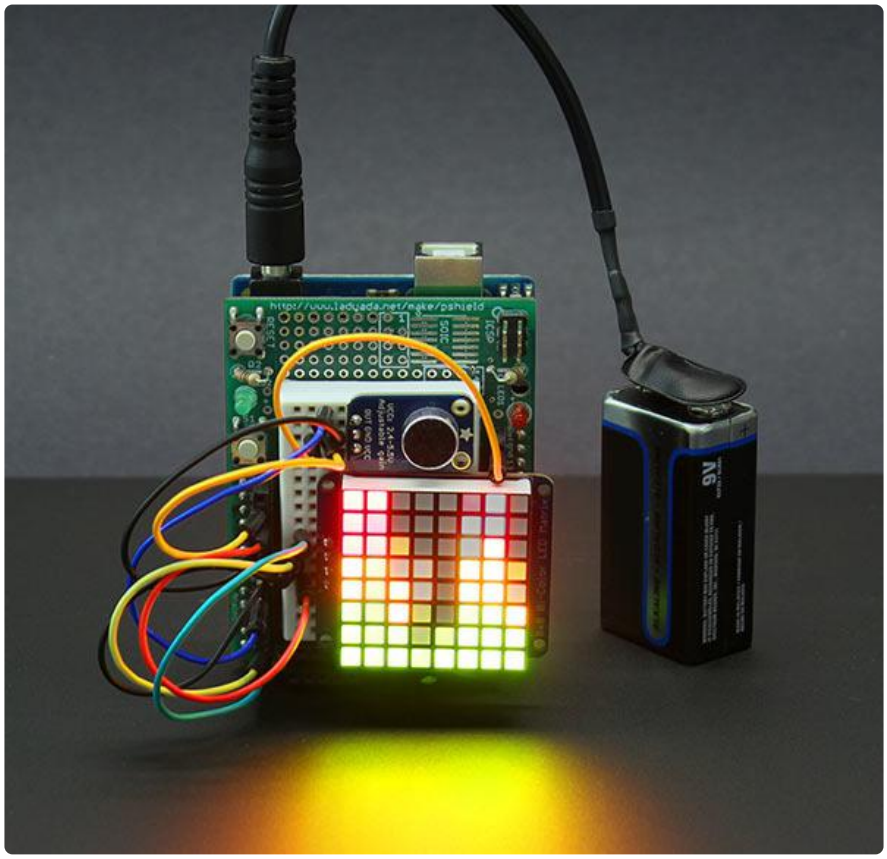
The raw audio samples are converted into a frequency spectrum using a fast Fourier transform () or FFT. There are a number of Arduino FFT libraries out there, but we keep finding ourselves returning to the venerable ELM-ChaN ffft library () for its speed

and good looks.

The FFT output still needs a bit of massaging to make for a good presentation on the limited 8x8 matrix. Several tables of scales and weights de-emphasize certain frequency ranges as they're reduced to just eight columns. The software works at keeping the graph interesting, but some columns will always be less lively than others, especially comparing live speech against music of varying genres. If everything seems to stick toward one end of the graph, try another musician, musical genre, or different speakers.

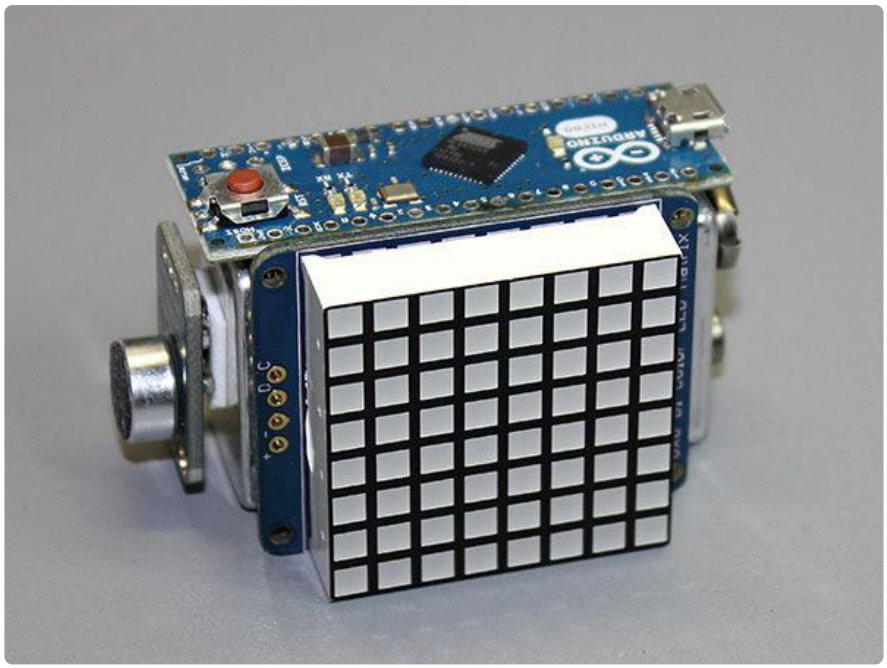
Ideas

Our wiring diagrams illustrated this project with a half-size breadboard, but for a smaller package the parts can also fit on an Arduino breakout shield:



Depending how you choose to orient the LED matrix, the matrix.setRotation() function can be used to keep the graph upright.

Just a concept — this is not functional — but it's interesting to note that an Arduino Micro, the LED matrix and the microphone amplifier are almost perfectly matched to the dimensions of a 9 Volt battery:



If you do attempt an extra-compact build like this, please keep in mind that the Arduino Micro includes a 5 Volt regulator, allowing it to be powered from a 9 Volt battery. Most small form-factor Arduino-compatible boards do not include a regulator, and will be damaged if you attempt to power them directly from 9 Volts! For such boards, either add your own regulator to the circuit, or use an appropriate battery pack: three alkaline AA or AAA cells in series, or four rechargeable NiMH cells.

Additionally, the code may need adjustments for some “alternative” Arduino boards… most likely the analog channel number, which may be mapped to different pin numbers on different boards.

Finally, because ADC registers are accessed directly, specific interrupts are used, and the FFT code is in AVR assembly language, this software will not run on upscale boards like the ESP32, Pico RP2040 or Teensy 3.0. It is strictly for “classic” Arduinos.