Sound Re-Ak-Tor

One of the sensors in your box is a sound detector. There are various types of microphone and sound detection devices available. The particular one we are using is a simple digital sound detector, sending a pulse when it ‘hears’ a sound.

The device has only 3 pins. Two are obvious, the OUT pin is pulled HIGH when a sound is detected. Onboard the device is a blue *trimpot*, which can be used to adjust the amplitude of sound at which the device triggers the OUT pin (note, not the frequency.) There are also 2 green LEDs on the device, the one on the VCC side simply indicates power, and the one on the OUT side lights in conjunction with the OUT pin being set.

In order to respond to sound programmatically, all we need to do is connect the OUT pin to any pin on our microcontroller and then read from that pin. If it is HIGH, we know a sound has been detected.

Given that, you should already see how it would be fairly easy to, say, make an LED flash when a sound is heard — just read from the device and write to a pin connected to an LED. But that’s not terribly interesting and the device itself already does that.

Instead we are going to combine it with the LED Strip to make a slightly more sophisticated light show. One where a varying number of LEDs will light depending on the relative ‘volume’ of sound. Maybe we can change color a bit as well.

Our task is to get some sort of ‘volume’ level from a device which only sends a single trigger when it detects sound of a certain amplitude. We will attempt to accomplish this using a technique called sampling. Most sounds vary in both frequency and amplitude over time, usually quite rapidly. If we simply check our sensor many times per second, and count how many times during that interval the trigger was HIGH, then we’ll have a value that vaguely represents how ‘loud’ it is!

This is related to PWM in a way. Instead of toggling an output rapidly to generate a fractional pseudo-analog value, we’re checking a digital value rapidly to arrive at a pseudo-analog value.

Although in our experiment we are only sampling the overall amplitude of sound, by sampling both amplitude and frequency any sound can be captured digitally for later playback. Virtually all modern recordings, telephone conversations, etc. are sampled and stored/processed/transmitted digitally.

In order to accurately reproduce the analog sound waves, sampling must be done at a high enough rate. Previously we acknowledged that human hearing is in the range or 20 - 20,000 Hz. The Shannon-Nyqvist theorems indicate that in order to accurately sample a waveform, samples must be taken at least twice the frequency of the wave being sampled. So, to sample music, a minimum of 40,000 samples per second are needed.

That certainly sounds like a lot, but for a modern computer it is not. Our little Arduino Nano has a clock rate of 16 MHz, each instruction takes 1-5 cycles, so even if it takes several instructions to capture a single sample, it has more than 10x the speed needed.

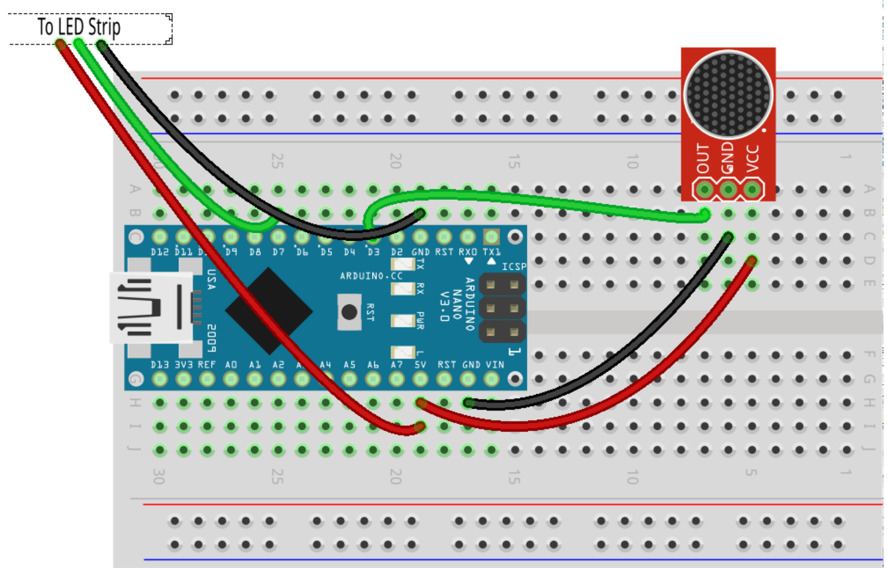
However, a 40KHz sample rate does wind up with a lot of data. Even if we only needed one byte per sample, that’s 40K bytes of memory per second, and the Nano has only 2K. We could only sample 0.05 seconds of sound! This mirrors the history of digital audio, sampling was easily possible with computers from the early 60s, but it wasn’t until the late ‘70s that the storage was available, and it took until the mid-90s before it was really practical and affordable. We could get the nano to do it by adding a SD card.

The program for this experiment is mostly complete, you will just need to understand its function and adjust the various parameters to your taste and to your device. As time permits, feel free to alter the math, colors, speed, etc. and see if you can produce some unique and interesting effects.

Note that the sound sensor is often more responsive to low frequencies than high — so a knock on the table is more likely to register than a clap or a whistle.

The name of this experiment is a play on the Volkswagen ‘soundaktor’, which is a computer controller noise maker used to ‘enhance’ engine sounds in various vehicles. Most enthusiasts disable or remove them, preferring on the real sounds.

Experiment: SoundReAktor

Components	Wiring Diagram
✓Microcontroller ✓Breadboard ✓RGB LED Strip ✓Sound Detector	

Connection Instructions

Provide power and ground from the 5V out of the Arduino to both the red & black lines of the LED strip and to the VCC & GND pins of the Sound Detector. Connect the green signal line of the LED strip to D7 of the Arduino. Connect the OUT pin of the Sound Detector to D3 on the Arduino. Adjust the trimpot such that the LED by the OUT pin is mostly off, but flickers on as you make noise near the device.

Sketch(es)	soundReAktor.ino

Analysis Questions

What is the purpose of the fadeToBlackBy() function call in this program.
Were you able to observe that enabling the debugging statements changes the resulting LED display?
Can you explain why?
Does changing the update frequency (set by the parameter passed to EVERY_N_MILLISECONDS) alter the display?
Can you explain why?

Programming Tasks

Adjust the values and mechanisms used to determine the number of LEDs to illuminate, so that an appropriate number of LEDs light for the overall sound in the room. Ideally speech nearby will illuminate around 1/2 and loud clapping will light them all. You may need to adjust the trimpot in conjunction with changing the various other parameters.

Similarly adjust the colors, you can change the formulae and the array of colors. Refer to the FastLED docs to find available pre-built color options.

Experiment with enabling and using the provided debugging print statements.

Notes

‘Trimpot’ is a name used for a small simple potentiometer used in a fixed circuit so that some small adjustments can be made. Trimpots can be used as voltage dividers or just variable resistors. They often have a more limited range and simpler internal construction than a full size potentiometer, and are generally intended to only be turned a few hundred times. Trying to use one as, say a volume switch, will likely result in premature failure.