Making noise with a SN76489 Digital Sound Generator – Part 3

By marco_c < https://arduinoplusplus.wordpress.com/author/arduinoplusplus/>

November 3, 2019 < https://arduinoplusplus.wordpress.com/2019/11/03/making-noise-with-a-sn76489-digital-sound-generator-part-3/>

In the first part < https://arduinoplusplus.wordpress.com/2019/10/05/making-noise-with-a-sn76489-digital-sound-generator-part-1/> and second part < https://arduinoplusplus.wordpress.com/2019/10/19/making-noise-with-a-sn76489-digital-sound-generator-part-2/> we examined the basics of the SN76489 hardware and the development of a library to manage sound production from this IC.

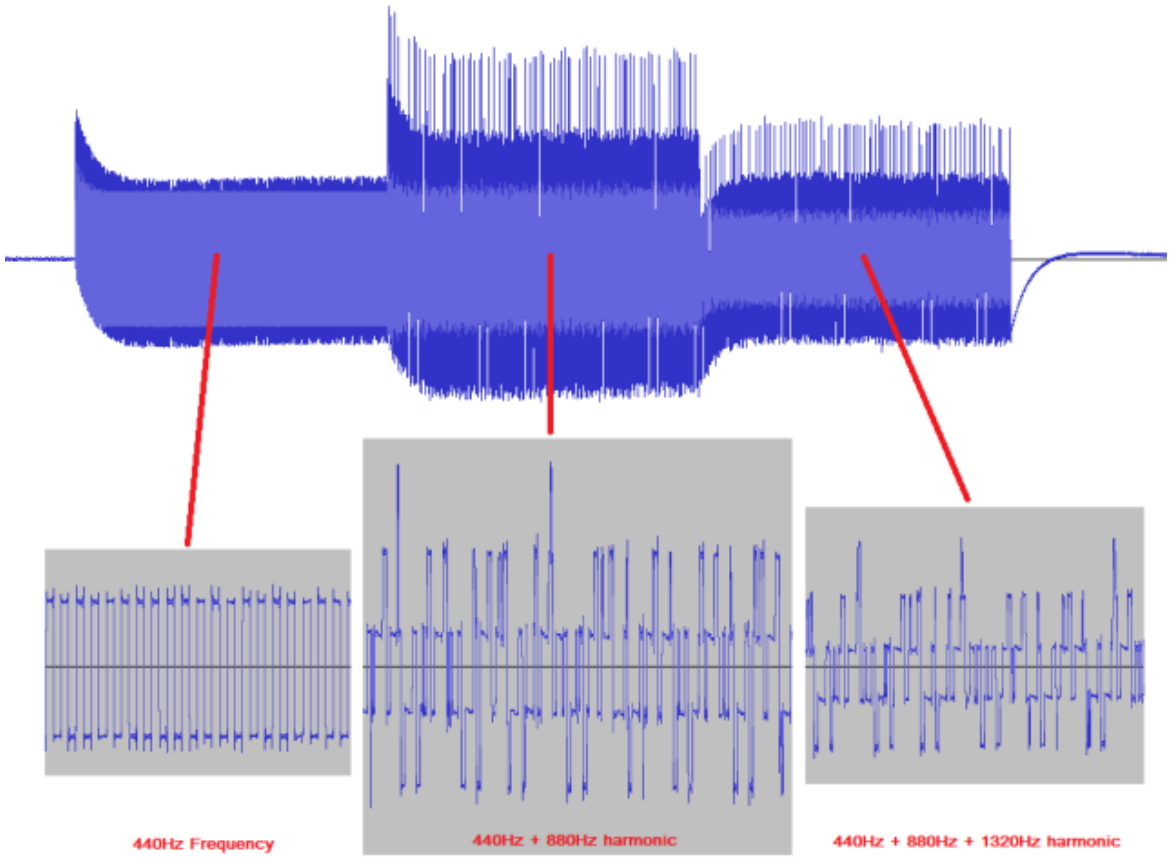
So, after all this effort, what kind of sound does this hardware produce? In this final post I run a few tests and dig into the resulting waveforms.

Tones and Harmonics

The IC produces a square wave, so any sound it produces will be a bit 'buzzy' as a pure tone. We can, however, make it sound a bit better by using the second and third sound channel to output 2x the frequency and 3x the frequency at a lower volume. This tries to imitate how harmonics are created from a real instrument. This is how it sounds (click

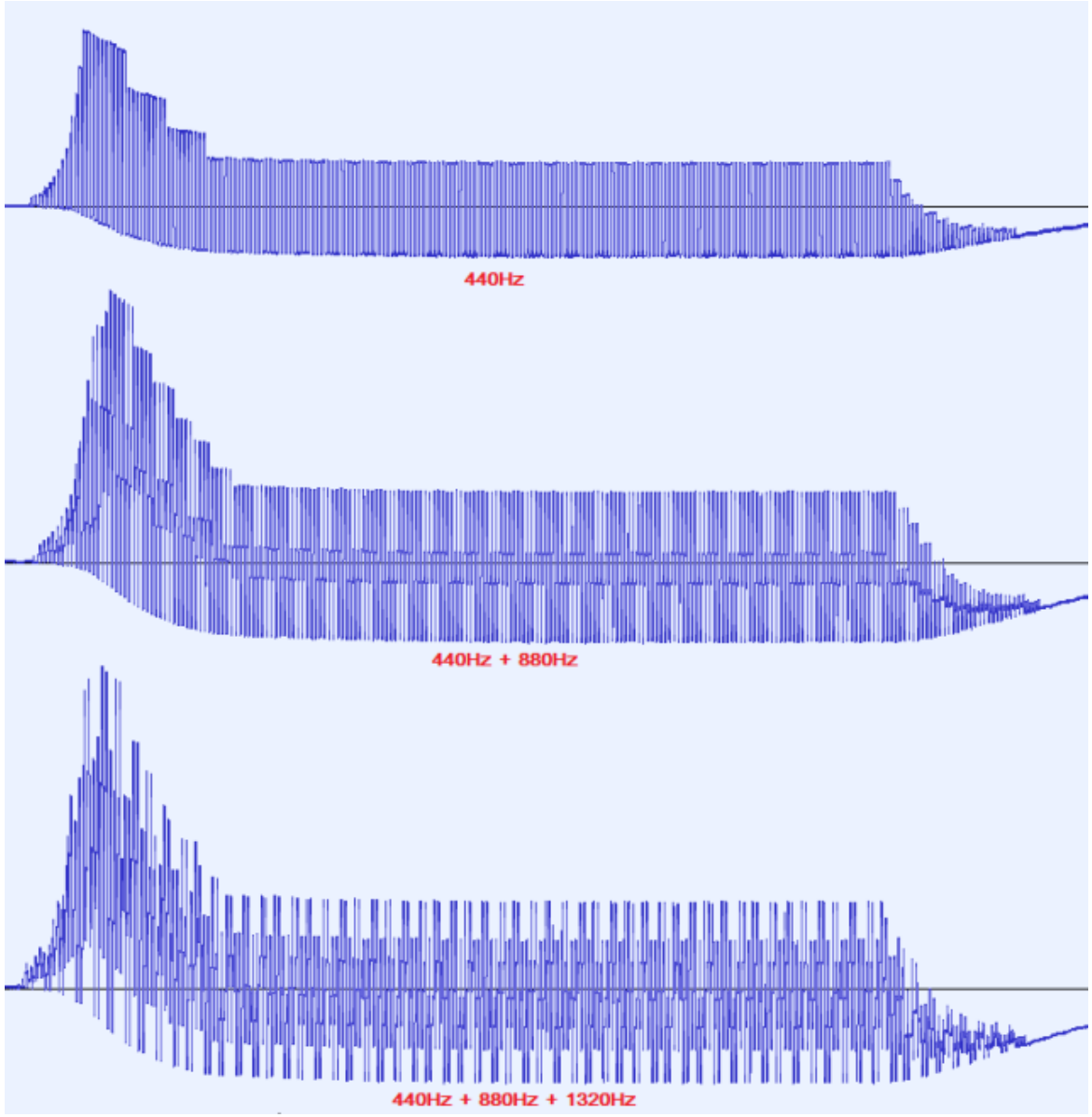
here to listen < https://www.dropbox.com/s/uu9nkvdkre6i44f/SN76489_Tone_440Hz.mp3?dl=0>) as a pure tone and with harmonics.

The captured waveform for that soundbite clearly shows the square waves (first section) and then more complex waves as the harmonics are added in. Interestingly the overall volume seems to drop when the third harmonic is added, and there is little difference in the quality of the sound with it included.



Adding ADSR

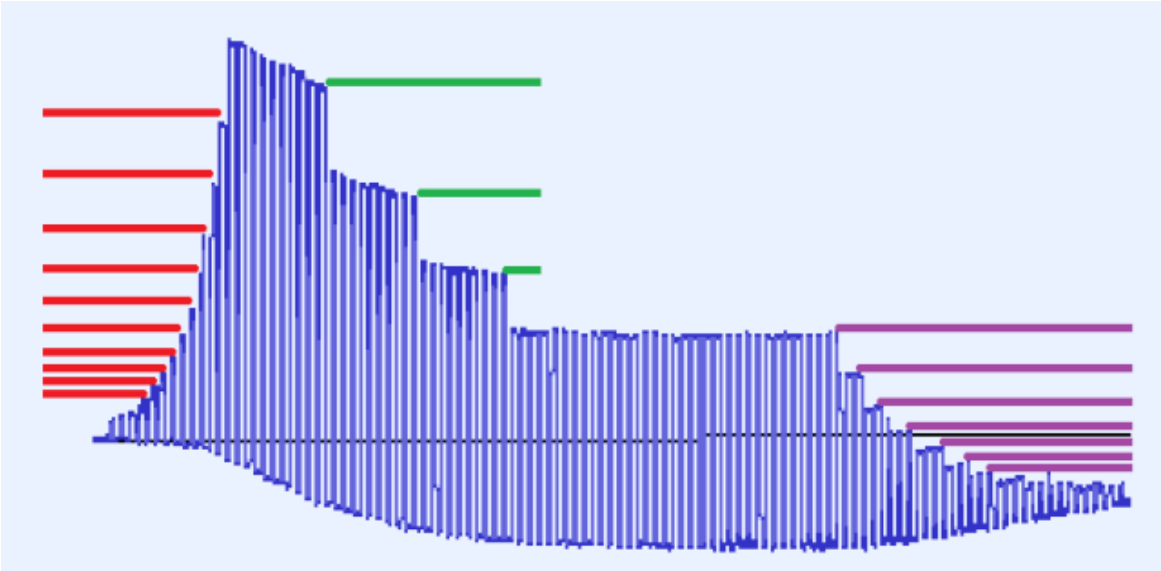
Adding ADSR (ie, playing a *note()* instead of a *tone()* using the library) creates a much better experience. The same note was played for 750ms using the pure frequency and with the additional harmonics. This is how it sounds (**click here to listen < https://www.dropbox.com/s/5tizh9k22fufwgc/SN76489_Note_440Hz.mp3?dl=0>**), and the waveforms are shown in the figure below.



< https://arduinoplusplus.files.wordpress.com/2019/10/sn76489-_note_440hz.png>

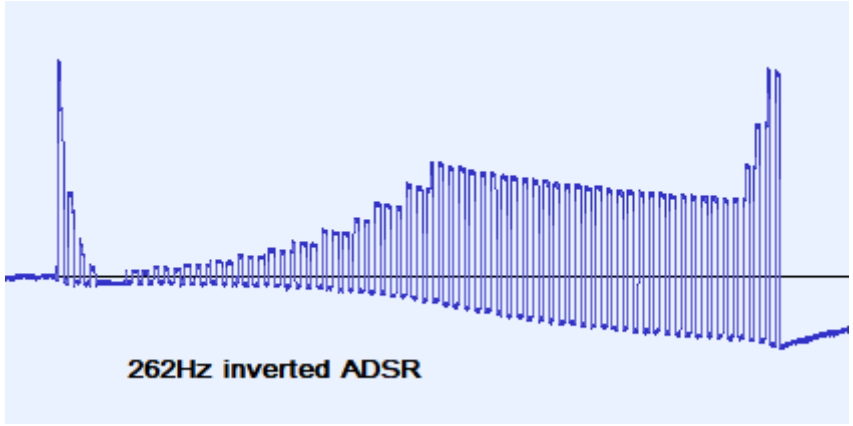
The three notes clearly sound different, and the third harmonic does make a difference in this case.

A closer inspection of the ADSR envelope is also interesting because we can clearly observe the points at which the volume changes during the A, D and R phases (red, green and purple lines in the figure below). Although the progression through the IC's attenuation levels is linear over time, the resulting waveform for A and R (with relatively short time base) are clearly exponential. This is because the ICs attenuation levels are given in dB, which is an exponential unit of measure to begin with.



< https://arduinoplusplus.files.wordpress.com/2019/10/sn76489-_actual_asdsr.png>

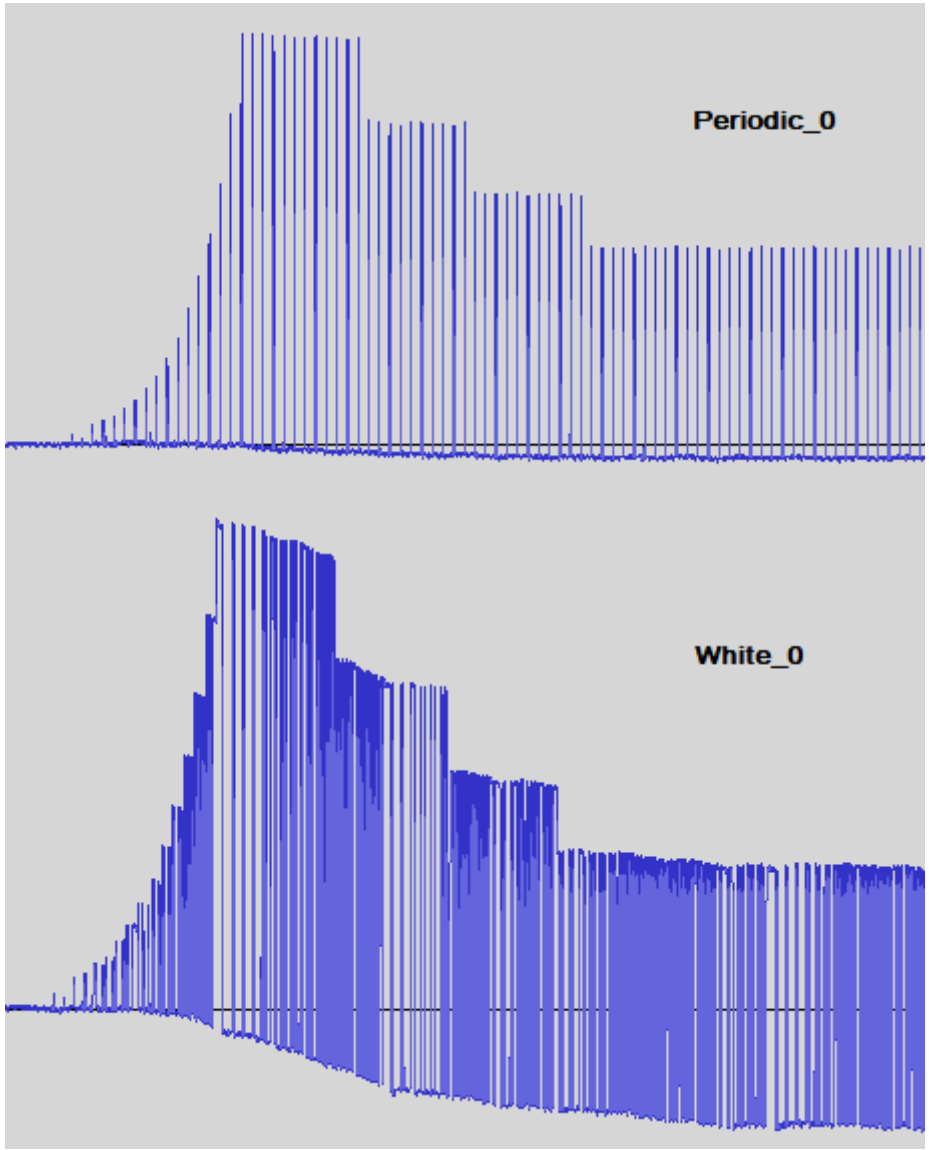
And, with an inverted ADSR envelope, the sound wave looks like this. Not that interesting, but good for contrast!



Periodic and White Noise

The SN74689 can also output 4 variants each of Periodic and White noise. I did not spend much time exploring this aspect of the IC, but I can see how percussion instruments like drums could be constructed from these kinds of sounds.

Periodic and White noise sound quite different (**click here to listen < https://www.dropbox.com/s/nwcsdhrianjsvtv/SN76498_Noise_P0123W0123.mp3?dl=0>**) and look different, as shown in the figure below.



One aspect that still puzzles me is how the type 3 noise generation interacts with the frequency generated by the tone generated by channel 3. Nothing much seemed to

make a difference to this and I could not find anything on the internet that explained it other than from a technical perspective. Perhaps an avenue for 'play' in future.

Playing RTTL files

Ring Tone Text Transfer Language (RTTTL) was developed by Nokia to define ringtones in cellphones. In moving on from playing single notes, RTTTL files provide an easy way evolve into playing musical 'tunes' .

The internet has a number of sites dedicated to RTTL music, with tens of thousands of RTTL string readily available for download (for example, see **this site < http://www.picaxe.com/RTTTL-Ringtones-for-Tune-Command/>**).

RTTL strings are very simple to decode and play, as they reflect the limitations of cellphone technology at the time the format was invented. An RTTL tune can be played with just one sound channel.

The RTTTL string defines the song metadata and the notes to be played. The string is divided into three specific sections – name, default value, and data – separated by a colon. The full format definition can be found at this **Wikipedia page < https://en.wikipedia.org/wiki/Ring_Tone_Transfer_Language>** . An example of an RTTL string is

```
fifth:d=4,o=5,b=63:8P,8G5,8G5,8G5,2D#5
```

One of the library example files is a RTTTL player and it can play RTTTL tunes with or without harmonics enabled. The difference in the how the notes sound is quite noticeable.

RTTL played with tone only (click to play). <https://www.dropbox.com/s/4hiscep3su1hoy4/SN76489_RTTL_Tone.mp3?dl=0>

RTTL played with harmonics (click to play). <https://www.dropbox.com/s/0p641yvcy554t8e/SN76489_RTTL_Overtone.mp3?dl=0>

Playing MIDI files

I adapted the MIDI file player in the **MD_MIDIFile library <https://github.com/MajicDesigns/MD_MIDIFile>** to use the SN76489 IC as the output device.

Very simple MIDI files (a single or 2 channels) play quite well and sound good. However, it quickly became evident that 3 channels did not work for most MIDI files, which generally expect more than 3 (usually 8-24) sound voices to be available.

I implemented a scheduling algorithm that used any idle SN76489 note channel for any requested MIDI channel, and it made some difference but still did not produce acceptable performance. Ganging up multiple SN74689 could provide a viable solution, but that was getting beyond the scope of what I wanted to do at this stage.

Conclusion

The SN86489 IC is a good way to add simple sound or musical effects to an Arduino project, although a simpler way for 'canned' mp3 files would be to use something like the YX5300 module (see this **previous article < https://arduinoplusplus.wordpress.com/2018/07/23/yx5300-serial-mp3-player-catalex-module/>**).

I came away from this exercise with a renewed respect for those original game developers who were able to produce tools for musicians to produce great chiptunes from this relatively simple device.

Up ↑