RESOURCES

ILLUMINATING MUSICAL CODE

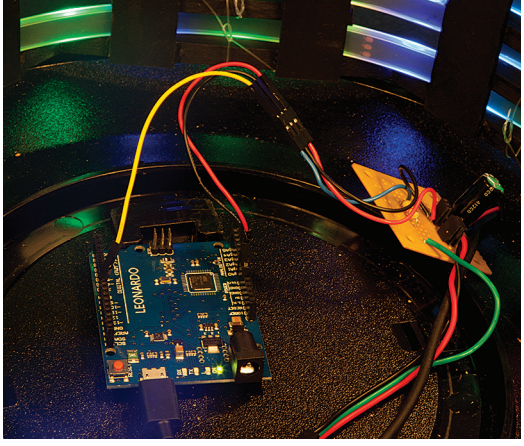
PROGRAM AN ELECTRONIC MUSIC PERFORMANCE IN REAL TIME

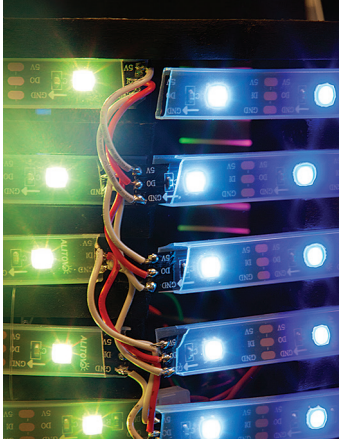
RESOURCES_HANDS ON

LIVE CODING IS A TYPE OF PERFORMANCE ART IN WHICH the performer creates music by programming and reprogramming a synthesizer as the composition plays. The synthesizer code is typically projected onto walls or screens for the audience to inspect as they listen to the unfolding sound. Live coding events are sometimes known as algoraves, and at these events it's common to see visualizations of the evolving music projected alongside the code. Often, these visualizations are created by a second performer manipulating graphics software in tandem with the live coder. ● After attending a few algoraves in New York City (musically, the results tend to fall along a spectrum from ambient soundscapes to pounding electronic dance music, with a few detours into more experimental domains), I decided to look a little closer at the software the performers were using. I wanted to see if I could come up with my own hardware spin on creating visualizations. While I'm not yet ready to take to the stage, the results have been fun. I'd recommend that any reader interested in music or sound art should try live coding, even if they have no experience playing any traditional musical instrument. ● The most popular software for live coding appears to be Sonic Pi. This is an open source project originally created by Sam Aaron for the Raspberry Pi, although it is also available for Windows and macOS. Sonic Pi's basic interface is a text editor. Apart from some performance-specific buttons, such as for starting and stopping a piece of music, it looks pretty much like any integrated development environment (IDE), in this case for a version of the Ruby language. Like Python, Ruby is an interpreted language that can run interactively. The Ruby-powered Sonic Pi IDE provides a friendly front end to the powerful SuperCollider sound-synthesis engine, which has been used for over two decades as the basis of many electronic music and acoustic research projects.

You *could* create a piece of music by typing a complete list of notes into the IDE, selecting a software-defined musical instrument plus any desired effects, such as reverb, and just having Sonic Pi play the tones. But this would eliminate the fun at the heart of live coding, which is a collaboration between the performer and the computer, in which the performer continually shapes algorithms

ERICA SNYDER (3)





THE GREAT PRETENDER: An Arduino Leonardo [above] acts as a USB device mimicking a MIDI-enabled electronic instrument. It converts received notes into colors displayed on a strip of LEDs [right].

but leaves the work of actually determining what note to play next up to those algorithms. Sonic Pi takes care of keeping everything in sync so that the music never misses a beat.

The most recent major version of Sonic Pi introduced the ability to send and receive MIDI messages. MIDI is the venerable standard used to communicate between computers and electronic instruments. In MIDI, notes are represented by a number from 0 to 127, with notes 21 to 108 covering the range of a grand piano. Originally, MIDI required a dedicated hardware interface, but today it's quite common to see MIDI being run over USB connections.

The addition of MIDI allowed me to press-gang some hardware to visualize the music produced on the fly by Sonic Pi. A while back, I had arranged 160 programmable WS2812B RGB LEDs in five tiers, so that they act like a 32- by 5-pixel color display. I built the display on a hexagonal wooden frame and mounted it in an empty "hat box" container once used to store removable disk packs. Not only does this upcycling allow me to justify hanging onto a bulky souvenir of a bygone technology, but the roomy inside of the box allows me to hide the frame and supporting electronics, in this case an Arduino Leonardo microcontroller. The Leonardo perfectly mimics USB devices, and I've used it before to make a custom controller for a spaceflight simulator. To drive so many LEDs, I added a 10-ampere power supply, with the power and USB cables running through a small hole I cut in the box's base.

I'd already used the Arduino MIDI library, which supports MIDI over a USB, at a music hackathon where I'd converted my hat-box display into a simple light organ. I could play a MIDI file from a computer and have the display change color according to the note. But my color mapping between note values and LED colors was quick and dirty to say the least: The same color was evoked by different notes.

For my Sonic Pi visualizer, I programmed the Leonardo using the FastLED library for both performance reasons and because of its support for the HSV (hue, saturation, value) color model. Mapping a value—such as a MIDI note—to a triplet of conventional RGB values is not straightforward, especially if you want all the notes to look equally bright. In contrast, with the HSV model, it's trivial to map a note to the hue byte while keeping the saturation and value bytes fixed.

Connecting the hat-box visualizer to the Sonic Pi software was an unremarkable if fiddly voyage through the various MIDI settings on my laptop. Sending a note to be visualized does require some changes to my Sonic Pi live code, however: As each note is generated algorithmically, I capture it using an intermediate variable rather than playing it immediately in a sound-synthesis instruction as I normally would. I use the intermediate variable to send the note to the hat-box display, via the "midi_note_on" command, in addition to playing the note audibly. This allows me to program the visualizer as I program the sound code.

My next step will be to program the hat box to respond to a set of custom MIDI control commands, which will allow me to alter how notes are mapped to hue values, or even select different visualization styles, on the fly. Then you might actually find me taking to the stage. **—STEPHEN CASS**

↗ **POST YOUR COMMENTS** at https://spectrum.ieee.org/sonic0919

RESOURCES_Q&A

HOW THE RASPBERRY PI INFILTRATED INDUSTRY

THE $35 COMPUTER'S CREATOR EXPLAINS WHAT DROVE THE LATEST REVISION

Seven years ago, Eben Upton created the first Raspberry Pi. As Upton told *IEEE Spectrum* in our March 2015 cover story, the Pi was inspired in part by his childhood experiments with a BBC Micro home computer: He wanted modern kids to have a simple machine that allowed for similar experimentation. Since then, the Pi has exploded in popularity, and the fourth major revision of the Pi was released in June. Upton talked with *Spectrum* senior editor Stephen Cass about the Pi 4's design, its growing commercial use, and what might be next.

Stephen Cass: How has the Pi's user base evolved?

Eben Upton: Our first year, our volume was almost entirely bought by hobbyists. But you have a lot of hobbyists who are also professional design engineers, and when their boss asked them to do something, often they used a Pi. So now you have people who are building industrial products around the Pi to resell. And then you have what we call, for want of a better word, DIY industrial, which is "I own a factory and I need control computers." And where I might have historically gone and bought an embedded PC, I'll buy a Pi. Last year we sold 6 million units and [we think as much as] half of those went to some kind of commercial use.