adafruit learning system

Bringing Back THE VOICE of Speak & Spell

Created by Phillip Burgess

Last updated on 2019-06-19 06:35:32 PM UTC

Overview

This project makes an **Adafruit Trellis M4** faithfully replicate the voice of *Speak & Spell.*

The **Texas Instruments Speak & Spell** and its descendants were electronic educational toys sold from the late 1970s into the early 1990s. If you were around for it, chances are **you *know* that voice**, even if you don't recall the spelling games that went with it.

Like any of our senses, sound can summon strong nostalgic feelings. The **Yamaha DX7** synthesizer left its fingerprints all over **1980s pop music**. Chiptune aficionados will *insist* that analog peculiarities of a real **Commodore 64's SID chip** can never be perfectly emulated in software. **Stephen Hawking** was so known for his specific voice that he kept (or emulated) it even as better-sounding technology came along.

And so it goes with early speech synthesis technology. A *Votrax SC-01* (used in MicroVox and Heathkit HERO 1) was distinct from *S.A.M. (Software Automatic Mouth)* was distinct from the General Instrument *SPO256* IC (which could be had from Radio Shack) and so forth. Each has its own peculiar *accent*. I can't explain *why* I wanted to **bring back the Speak & Spell voice** specifically. I just...did. It's that nostalgia thing, and anybody who knows and hears it *will* laugh. That's really why I do any of this stuff.

A new "retro" Speak & Spell is being released and I was disappointed that its voice is modern and understandable. That's like when a cartoon voice actor is replaced and anyone can spot the change.

Software

Ready-Made Software

The example software is designed for the **Adafruit Trellis M4** board (though the underlying speech functions can be used with other boards...see “Building From Source” below).

Plug a **Trellis M4** into your computer with a **USB** cable, then **double-click the RESET button** on the back of the board.

After a moment, a small flash drive called **TRELM4BOOT** should appear on your system. Drag-and-drop the **.UF2 file** (downloadable below) on to this flash drive. You may briefly see a file-copying dialog, and then after a few seconds the drive will be ejected and you’ll get a rainbow row of lights on the Trellis.

https://adafru.it/F3f

https://adafru.it/F3f

This will overwrite CircuitPython if that’s what’s currently on your Trellis board. Your CircuitPython *code* is **intact**, only the *interpreter* has been replaced with this project. You can **restore** CircuitPython by downloading the .UF2 file here (https://adafru.it/Em6).

If you press the reset button only once you may get a drive in your computer's file explorer/finder named CIRCUITPY. Click the reset button twice (it may take practice) to get it to the TRELM4BOOT drive - you want to put your file there.

Triggering Sounds

Connect **headphones** or a powered **speaker** to the audio output of the Trellis M4.

This code doesn’t play any of the original’s spelling games, it’s strictly a “**sound board**.”

The **bottom row** of buttons on the Trellis M4 will be lit a rainbow of colors. Think of these like the “shift” or “control” keys on your keyboard...they **modify** all the other buttons. **Hold down** one of these buttons and the rest of Trellis will light up that same color. Now, still holding that shift button, you can **tap** any of the other buttons to trigger different

words, phrases, letters or numbers. Additionally, the top three rows trigger sounds when no shift button is held. That's 272 authentic Speak & Spell noises in total!

Other Ways to Experience "The Voice"

If you don't have a Trellis M4 and don't want to get into adapting the code, there's a fully emulated version of Speak & Spell playable in your web browser (https://adafru.it/F3y) (seems to work most reliably with Google Chrome).

What's fascinating about this site is that it's *not* simply a reproduction playing back canned WAV samples (though one or two of those exist online as well)...it's actually fully **emulating** the Speak & Spell hardware and speech synthesis chip on the fly, and you can play the original spelling games if you want.

Of course you can always just pick one up on eBay. They're not horribly expensive if you don't mind one with a few scuffs. The fun of this Trellis project though is that it lets you string together *all the sounds* as you want, use them in new ways, or as a musical instrument (much more easily than "circuit bending" the original toy).

Building From Source

This Trellis sound board is an **Arduino** project and **the source code can be downloaded from Github (https://adafru.it/F3A)**. There are two files: `TalkieTrellis.ino` (which handles button presses and triggers sounds) and `words.h` (containing audio data the same way it was represented by Speak & Spell — these are *not* simply WAV samples).

Several prerequisite libraries must be installed. This can be done through the Arduino Library Manager (Sketch→Include Library→Manage Libraries...). Search for:

- Adafruit_NeoTrellisM4
- Adafruit_Keypad
- Adafruit_NeoPixel

Additionally, the Adafruit fork of the **Talkie** library must be downloaded and installed **manually** (it's not in the Library Manager):

- Talkie library on Github (https://adafru.it/F3B)
- How to manually install an Arduino Library (https://adafru.it/m3e)

The Adafruit fork works on **SAMD** microcontrollers and adds support for emulating the Speak & Spell's **TMS5100** speech chip (the original Talkie lib emulates only the TMS5220, a later speech chip used in the TI99-4/A and others).

Though our example sketch is specific to the Trellis M4, the Talkie library itself should work with pretty much any AVR or SAMD board! Even a classic Arduino Uno, using an Adafruit Wave Shield, or with a piezo speaker between pin 3 and ground. On SAMD boards, pin A0 provides true analog output.

Several examples included with the Talkie library demonstrate its use. These are all emulating the **TMS5220** but the basic idea is the same: there are some tables of speech data (usually extracted from some vintage device's ROM) and one then calls the library's `say()` function, passing in a pointer to different data for different words...for example, these lines from the clock demo speak the word "six":

```
const uint8_t spSIX[] PROGMEM = {0x0E,0xD8,0xAE,0xDD,0x03,0x0E,0x38,0xA6,0xD2,0x01,0xD3,0xB4,0x2C,0xAD,0x

voice.say(spSIX);
```

Our Trellis code just does one thing differently...the Talkie library must be *explicitly switched* to **TMS5100** speech mode before using any Speak & Spell dialogue:

```
voice.mode(TALKIE_TMS5100);
```

Copy and paste encoded words and phrases from the *TalkieTrellis* sketch (in the words.h file) into your own code.

Speech data for the TMS5100 and TMS5220 is *not* cross-compatible. If you want to use both in the same sketch, it's necessary to call `voice.mode(TALKIE_TMS5100)` before using any TMS5100 words, and `voice.mode(TALKIE_TMS5220)` before any TMS5220 words.

How it Works

As a kid in the 1980s, there was something of an **urban legend** around the Speak & Spell that went by at least two variations:

- That there was some **magical button combination** that would unlock **dirty words**.
- Or — if you kept **typing** dirty words on a Speak & Spell, that it would **scold you**.

Though nobody ever actually *heard* a Speak & Spell do these things firsthand, such rumors persisted even to the present day*. But...understanding a bit how the TMS5100 speech chip works...this myth is totally **busted**.

* Much like the fabled "chemical that turns purple if someone pees in the pool," also nonsense.

Most speech synthesizers of the day were based on *phonemes* — the smallest pieces of speech — which could be pieced together to form words and sentences. This gave them essentially an *unlimited* vocabulary, but the downside is a robotic **monotone voice**. Some tools could vary the speed or pitch somewhat, but at best these sounded like the Muppets' Swedish Chef.

The standout feature of these T.I. speech chips was that they instead used *linear predictive coding,* a highly compressed lossy audio format[1]. Being based on actual **recorded speech**, a person's unique timbre, inflection and even accents are possible (Speak & Spell toys released in a few other countries were voiced by native speakers...the American English item is distinct from British English, for example[2]). But this also means they *can't* say anything willy-nilly...if it's not recorded and stored in the ROM, it's outside the chip's vocabulary[3]. With folks having picked through every byte of the Speak & Spell ROM with a fine-toothed comb...we now know every word and phrase that's in there, and that there are *no* swear words, nor any scolding[4].

1. And I do mean *lossy.* About 300 recorded words and phrases — a few minutes' worth — fit in the Speak & Spell's tiny **32 kilobyte** ROM. The format — called *LPC-10* — was part of a Federal standard (https://adafru.it/F3P) for voice communications with limited bandwidth. A later variant of linear predictive coding is used in GSM cell phones...if you've ever heard someone getting out of range and their voice "sounds like a badly-compressed JPEG looks," that's a form of LPC struggling with fewer and fewer bits.
2. There are tales of certain words in different Speak & Spell models where the original voice talent was not around to record changes...so you'll get these one-off words that were instead spoken by an engineer...or in some cases *hand-editing* LPC tables (https://adafru.it/F3Q) through trial-and-error.
3. Not *entirely* true. The TI99-4/A *Terminal Emulator II* module used a set of LPC-encoded phonemes coupled with a text-to-speech algorithm. But this is not how the Speak & Spell operated.
4. There is, however, *one* orphaned word in the Speak & Spell ROM: "mosquito." The LPC-encoded speech is present, but no spelling, nor is it linked to in any of the spelling word lists. Most likely it was decided that this was outside the target demographics' vocabulary...but I like to imagine that someone took the task of "debugging" too literally. The word is included in the TalkieTrellis sketch.

What makes the original Speak & Spell voice so distinctly *Speak & Spell* is that it's **one actual specific person's voice** —

Dallas TX radio announcer Mitch Carr (https://adafru.it/F3Z) for the U.S. model — not a piecing-together of synthetic phonemes. It then picks up an additional thick "technological accent" through the heavy processing of LPC encoding, storage and reconstruction.

T.I. had one (perhaps a few) special machine(s) that they would cart to these recording sessions and could perform the encoding and playback on-site. Probably long since dismantled.

Acknowledgements and Rabbit Holes

The **Talkie library** for Arduino was written by Peter Knight (https://adafru.it/F3R), using insights and data from the MAME emulator (credit to authors and helpers within the "tms" files here (https://adafru.it/F3S)).

Talkie originally emulated the TMS5220 speech chip, whereas Speak & Spell used its earlier sibling, the TMS5100. At first I thought it would be easiest to "rearrange" the TMS5100 speech data into the TMS5220 format and use that with Talkie. This is *somewhat* possible, but the result would incur noticeable shifts...it would not be a faithful reproduction, which was the *whole point* of this exercise. Going back to the MAME source code, it turns out the synthesis math is identical, and it's mostly a matter of different coefficient tables between the two chips. I brought the TMS5100 tables over from MAME back into Talkie, and with just some small changes it's now selectable between the two. (But it was a very roundabout journey getting there.)

Insights into the Speak & Spell ROM format — to extract all the words and phrases — came from furrtek.org (https://adafru.it/F3T). Their project was aiming to add *new* words to a Speak & Spell, but the info there was super helpful in getting the *old* data *out.*

Encoding new data into Talkie (or a real Speak & Spell) turns out to be *quite* a challenge. Not computationally — we have more power than we know what to do with sometimes — but that there's no longer any *readily-usable software* to perform the LPC-10 compression with the same sort of results that Texas Instruments achieved. That code is just *gone.* The nearest thing that's known is called *QBoxPro* — an *ancient* 16-bit Windows application. A link to software, and a description of how to use it on modern systems, are also present on the furrtek.org site (https://adafru.it/F3U).

Files

https://adafru.it/F3X

https://adafru.it/F3X

https://adafru.it/F3Y

https://adafru.it/F3Y