Trellis M4 Beat Sequencers
Created by John Park

https://learn.adafruit.com/trellis-m4-beat-sequencer

Last updated on 2023-08-29 03:55:08 PM EDT
Table of Contents

Overview 3
- About the NeoTrellis M4
- Parts

Eight-Step Drummer 5
- CircuitPython Prep
- Prepare Audio Files
- Sound Packs
- Samples Directory
- Copy Sounds
- Rename
- CircuitPython Code
- Sequence Tracking Bar
- Program Beat Patterns
- Kick
- Snare
- Hi-Hats
- "Cowbell"

Sixteen-Step Sequencer/Sampler 15
- Usage Overview
- Firmware
- Interface Tour
- Sounds
- Edit
- Transport
- Beat Repeats and Step Muting
- Tilt Effects
- Kit Solo
- Tempo
- Volume
- Inspiration
- Deeper Modification

FAQ/Troubleshooting 24
Drum machines make the music world go round. While originally intended to replace real, human drummers (which they never have, thankfully!) drum machines and sequencers played a huge role in creating whole new genres of music from hip hop to electronica, dub to disco.

Now, you can get started making your own beats using the Trellis M4 from Adafruit!

We'll start with a simple eight-step sequencer, programmed in CircuitPython, and then get a bit more complex with a sixteen-step sequencer that includes effects, beat repeats, and even a live sampler to record your own sounds.

Unlike a MIDI-based sequencer, these two sequencers are self-contained, requiring no external synth or computer. Simply plug in an 1/8" audio cable and all of the onboard sounds will play through your amplifier or headphones!

About the NeoTrellis M4

The NeoTrellis M4 is an all-in-one Audio board, ready to become your next synth, soundboard, drum machine, keyboard, or any other invention you'd like to adapt it for. It's powered by the SAMD51 microcontroller, a Cortex M4 core running at 120 MHz, featuring a roomy 512KB of flash and 192KB of SRAM. A separate flash chip provides a full 8MB of space for files and audio clips.
On the front side is a 4x8 grid of elastomer button pads with a NeoPixel nestled in the center of each one. You can read any/all button presses simultaneously thanks to the fully diode’d matrix, and also set each button color to any of 24-bit colors.

### Parts

<table>
<thead>
<tr>
<th>Part Description</th>
<th>Adafruit NeoTrellis M4 with Enclosure and Buttons Kit Pack</th>
<th>Adafruit NeoPotentiometer Adjustable Headphones</th>
<th>1/8&quot; TRRS connector earbuds with mic and slider</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>So you’ve got a cool/witty name for your band, a Soundcloud account, a 3D-printed Daft Punk...</td>
<td><a href="https://www.adafruit.com/product/3959">https://www.adafruit.com/product/3959</a></td>
<td><a href="https://www.adafruit.com/product/4020">https://www.adafruit.com/product/4020</a></td>
</tr>
<tr>
<td></td>
<td><a href="https://www.adafruit.com/product/592">https://www.adafruit.com/product/592</a></td>
<td>Standard A to micro-B USB cable - 3ft</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><a href="https://www.adafruit.com/product/2698">https://www.adafruit.com/product/2698</a></td>
<td>1 meter</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td><a href="https://www.adafruit.com/product/1363">https://www.adafruit.com/product/1363</a></td>
<td>USB Powered Speakers</td>
<td>Add some extra boom to your audio project with these powered loudspeakers.</td>
</tr>
<tr>
<td></td>
<td><a href="https://www.adafruit.com/product/1363">https://www.adafruit.com/product/1363</a></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Monoprice 5-Watt Guitar Amplifier, Portable Recorder

Always be ready rock out or record with this 5-watt Guitar Amplifier, Portable Recorder, and USB Audio Interface from Monoprice. This handheld 5-watt amp...

https://www.adafruit.com/product/4402

© Adafruit Industries
Stereo 3.5mm Plug/Plug Audio Cable - 6 feet
This basic cable comes with two 3.5mm (1/8" headphone jack size) stereo connectors. It's fairly straightforward, you'll commonly need these to connect two audio devices...
https://www.adafruit.com/product/876

Eight-Step Drummer
This sequencer is perfect for getting started! It's a lot of fun to play, and very intuitive, since all four sample tracks are displayed continuously. No "menu-diving" allowed!

You'll program it with CircuitPython, so first, let's prep the board for that.

CircuitPython Prep
To prep the Trellis M4 to run the sequencer code, follow these steps:

• Update the bootloader for NeoTrellis from the Trellis M4 guide
• Install the latest CircuitPython for NeoTrellis from the Trellis M4 guide
• Get the latest CircuitPython library pack, matching the version number with the major version of CircuitPython you are using, unzip it, and drag the libraries you need over into the /lib folder on CIRCUITPY. The latest library package includes support for Trellis M4.
https://github.com/adafruit/Adafruit_CircuitPython_Bundle/releases/
If you'd like to reduce the storage footprint of the lib folder (to make room for lots of sound files!) these are the only one's you'll need to keep, you can remove the rest:

- adafruit_trellism4.mpy
- adafruit_fancyled folder
- neopixel.mpy
- adafruit_matrixkeypad.mpy
- adafruit_adxl34x.mpy
- adafruit_bus_device

Prepare Audio Files

Audio files can be gathered by any means you like, but they will all need to be formatted the same way to be recognized by the Trellis M4.

See this guide on how to convert audio files.

For the Eight-Step Simple Sequencer you can use either mono or stereo files, just make sure all of your files are the same format, no mixing and matching!

In order to prevent the program from playing errant .wav files that may already be on the drive, delete any .wav files you may see left over from other projects on your CIRCUITPY drive before adding new ones.

Sound Packs

For your convenience and drumming pleasure, we've compiled some sound kits for you to use. Simply download this zip file, uncompress it, and you'll be able to use them on the board after copying and renaming them as outlined below.

Samples Directory

With your Trellis M4 plugged into your computer, it will show up as the CIRCUITPY USB drive. Open it up and create a new folder named samples.
Copy Sounds

Copy a set of four sound files from one of the packs and paste it into the samples folder you just made on the Trellis M4’s CIRCUITPY drive.

Rename

Rename the four files like this:

- drum_cymbal_pedal.wav to voice01.wav
- drum_snare_hard.wav to voice02.wav
- elec_ping.wav to voice03.wav
- bd_haus.wav to voice04.wav
As you’ll see in the code below, this name pattern: /samples/voice01.wav through /samples/voice04.wav is the only set of names that will be found and played.

When we code the Trellis M4 with the sequencer program, the voices (sound files) will be arrayed along the four rows as shown here.

CircuitPython Code

Everything is now prepared, you just need the actual code!
Copy the code shown here, and then paste it into Mu. Save it to your Trellis M4 as code.py

```python
import time
import board
import busio
import audioio
import audiocore
import audiomixer
import adafruit_fancyled.adafruit_fancyled as fancy
import adafruit_trellism4
import adafruit_adxl34x

tempo = 180  # Starting BPM

# You can use the accelerometer to speed/slow down tempo by tilting!
ENABLE_TILT_TEMPO = True
MIN_TEMPO = 100
MAX_TEMPO = 300

SAMPLE_FOLDER = "/samples/"  # the name of the folder containing the samples
# You get 4 voices, they must all have the same sample rate and must
# all be mono or stereo (no mix-n-match!)
VOICES = [SAMPLE_FOLDER+"voice01.wav",
          SAMPLE_FOLDER+"voice02.wav",
          SAMPLE_FOLDER+"voice03.wav",
          SAMPLE_FOLDER+"voice04.wav"]

# four colors for the 4 voices, using 0 or 255 only will reduce buzz
DRUM_COLOR = ((0, 255, 255),
              (0, 255, 0),
              (255, 255, 0),
              (255, 0, 0))

# For the intro, pick any number of colors to make a fancy gradient!
INTRO_SWIRL = [fancy.CRGB(255, 0, 0),  # red
               fancy.CRGB(0, 255, 0),  # green
               fancy.CRGB(0, 0, 255)]  # blue

# the color for the sweeping ticker bar
TICKER_COLOR = (255, 255, 255)

# Our keypad + neopixel driver
trellis = adafruit_trellism4.TrellisM4Express(rotation=90)

# Our accelerometer
i2c = busio.I2C(board.ACCELEROMETER_SCL, board.ACCELEROMETER_SDA)
accelerometer = adafruit_adxl34x.ADXL345(i2c)

# Play the welcome wav (if its there)
with audioio.AudioOut(board.A1, right_channel=board.A0) as audio:
    try:
        f = open("welcome.wav", "rb")
        wave = audiocore.WaveFile(f)
        audio.play(wave)
        swirl = 0  # we'll swirl through the colors in the gradient
        while audio.playing:
            for i in range(32):
                palette_index = (((swirl+i) % 32) / 32
                color = fancy.palette_lookup(INTRO_SWIRL, palette_index)
                # display it!
                trellis.pixels[(i//8, i%8)] = color.pack()
            swirl += 1
            time.sleep(0.005)
        f.close()
```
# Clear all pixels
trellis.pixels.fill(0)
# just hold a moment
time.sleep(0.5)
ext except OSError:
    # no biggie, they probably deleted it
    pass

# Parse the first file to figure out what format its in
with open(VOICES[0], "rb") as f:
    wav = audiocore.WaveFile(f)
    print("%d channels, %d bits per sample, %d Hz sample rate " %
        (wav.channel_count, wav.bits_per_sample, wav.sample_rate))

    # Audio playback object - we'll go with either mono or stereo depending on
    # what we see in the first file
    if wav.channel_count == 1:
        audio = audioio.AudioOut(board.A1)
    elif wav.channel_count == 2:
        audio = audioio.AudioOut(board.A1, right_channel=board.A0)
    else:
        raise RuntimeError("Must be mono or stereo waves!")

    mixer = audiomixer.Mixer(voice_count=4,
        sample_rate=wav.sample_rate,
        channel_count=wav.channel_count,
        bits_per_sample=wav.bits_per_sample,
        samples_signed=True)

    audio.play(mixer)

samples = []
# Read the 4 wave files, convert to stereo samples, and store
# (show load status on neopixels and play audio once loaded too!)
for v in range(4):
    trellis.pixels[(v, 0)] = DRUM_COLOR[v]
    wave_file = open(VOICES[v], "rb")
    # OK we managed to open the wave OK
    for x in range(1, 4):
        trellis.pixels[(v, x)] = DRUM_COLOR[v]
        sample = audiocore.WaveFile(wave_file)
        # debug play back on load!
        mixer.play(sample, voice=0)
        for x in range(4, 7):
            trellis.pixels[(v, x)] = DRUM_COLOR[v]
            while mixer.playing:
                pass
            trellis.pixels[(v, 7)] = DRUM_COLOR[v]
    samples.append(sample)

# Clear all pixels
trellis.pixels.fill(0)

# Our global state
current_step = 7 # we actually start on the last step since we increment first
# the state of the sequencer
# currently pressed buttons
current_press = set()

while True:
    stamp = time.monotonic()
    # redraw the last step to remove the ticker bar (e.g. 'normal' view)
    for y in range(4):
        color = 0
        if beatset[y][current_step]:
            color = DRUM_COLOR[y]
        trellis.pixels[(y, current_step)] = color

    # next beat!
current_step = (current_step + 1) % 8

# draw the vertical ticker bar, with selected voices highlighted
for y in range(4):
    if beatset[y][current_step]:
        r, g, b = DRUM_COLOR[y]
        color = (r//2, g//2, b//2)  # this voice is enabled
        #print("Playing: ", VOICES[y])
        mixer.play(samples[y], voice=y)
    else:
        color = TICKER_COLOR      # no voice on
    trellis.pixels[(y, current_step)] = color

# handle button presses while we're waiting for the next tempo beat
# also check the accelerometer if we're using it, to adjust tempo
while time.monotonic() - stamp < 60/tempo:
    # Check for pressed buttons
    pressed = set(trellis.pressed_keys)
    #print(pressed)
    for down in pressed - current_press:
        y = down[0]
        x = down[1]
        beatset[y][x] = not beatset[y][x] # enable the voice
        if beatset[y][x]:
            color = DRUM_COLOR[y]
        else:
            color = 0
        trellis.pixels[down] = color
    current_press = pressed

if ENABLE_TILT_TEMPO:
    # Check accelerometer tilt!
    tilt = accelerometer.acceleration[1]
    #print("%0.1f" % tilt)
    new_tempo = tempo
    if tilt < -9:
        new_tempo = tempo + 5
    elif tilt < -6:
        new_tempo = tempo + 1
    elif tilt > 9:
        new_tempo = tempo - 5
    elif tilt > 6:
        new_tempo = tempo - 1
    if new_tempo != tempo:
        tempo = max(min(new_tempo, MAX_TEMPO), MIN_TEMPO)
    print("Tempo: %d BPM" % tempo)
    time.sleep(0.05)  # dont update tempo too fast!
    time.sleep(0.01)  # a little delay here helps avoid debounce annoyances

Sequence Tracking Bar

When the board restarts, it will play through all four sound files, and then start running
the sequence tracking bar forward
There are many ways to conceptualize an eight step sequence. I like to think of it as a single measure with 4/4 timing, meaning there are four beats to the measure and a quarter note gets the beat. So, our sequence is made of eight eighth notes, and the steps can be counted as:

"one-and-two-and-three-and-four-and"

Program Beat Patterns

Now, you can start making up some beats using the samples!
Kick

Try pressing the first and fifth buttons on the bottom row to start up a kick drum on the first and third beats of the measure (remember, there's a beat on every other button pad, with syncopated subdivision between them).

Snare

Now we'll add in a snare on step 3 and 8, which are the second beat and the "and" after the fourth beat as shown here.
Hi-Hats

Now let's lay in hi-hats on nearly every beat, as shown. While the Trellis M4 can play multiple sounds simultaneously, sometimes patterns sound nicer when there isn't too much going on on every step!

"Cowbell"

Lastly, we'll throw in an electronic blip sound that's reminiscent of the "cowbell" on a Roland TR-808. We'll put it on the two and then on the eighth note divisions of the three and four.

Here is what the sequence sounds like with the 808 style voices:
And here it is at a higher BPM with some genuine Adafruit warehouse sounds! You can increase and decrease the tempo just by tilting the Trellis M4 to the right or left. The Analog Devices ADXL343 accelerometer built right on the board detects the rotation and the CircuitPython code translates that data into tempo changes.

Sixteen-Step Sequencer/Sampler

This is a more fully featured sequencer, plus, it's a live sampler! It is a 16-step drum and voice/recording sequencer with eight built-in drum samples and eight slots for recording fresh, live samples through a plugged-in microphone.

Here are the key features:

- Sixteen step sequencer
- Two banks of eight sounds: a built-in drum kit and a sampled kit of your own devising!
- Step muting
- Beat repeats
- Tilt (accelerometer) effects
- Bank soloing
- Tempo up/down
- Volume up/down
- Endless fun!
Usage Overview

This will give you an idea of general usage, and then we'll go into more details and examples below.

You can create your own drum patterns by selecting one of the kit sounds and then, in write mode, "placing" that sound into any of the 16 sequence steps. You can repeat this with all of the different sounds you want to use, and multiple sounds can be played from the same sequence step.

Hit play at any time to hear your beats played over a set of headphones or powered speakers! You can even edit the pattern while playing, or, switch out of write mode and play live fills on top of the sequence.

You can add interest to your patterns by triggering three types of beat repeats, or, layer on some tilt effects that can be adjusted by simply tilting the Trellis M4 from side to side!

Each of the two banks of kit sounds can be solo'd, there are buttons to adjust tempo and volume, and you can mute any step in the sequence in real time during playback for breakdowns and build ups!

Firmware

All you need to do to use this sequencer on your Trellis M4 is to plug it into USB, download the firmware .uf2 file below, place your Trellis M4 into bootloader mode by double-clicking the reset button on the back, and drag the firmware file onto the TREL SM4BOOT drive that shows up.

This program will replace CircuitPython, to get back to your CircuitPython projects, reinstall it by following https://learn.adafruit.com/adafruit-neotrellis-m4/circuitpython

Interface Tour

Let's have a look at the interface of your sequencer. We use color coding to signify different groupings.
The top two rows are the sequence steps, which have a couple of different uses we'll cover in a bit.

The bottom two rows are broken up into the following groupings:

- edit
- beat repeat
- tilt effects
- kit solo
- tempo
- volume
- transport

Most buttons have a single use, which makes pretty easy to learn to program and play your beats, rather than having lots of "menu diving" and secret modes. That said, there are a few modes to enter using the edit buttons which will change the function of the sequence step buttons. We'll go over these in a moment.

Sounds

There are two banks of sounds available for sequencing and live play. The top row is the built-in kit, which has the following:

- kick 1
- kick 2
- clap
- crash
- closed hi-hat
- open hi-hat
- rim
- snare

The second row is the sampled kit. This is where you can record eight of your own samples using a microphone on a set of earbuds plugged into the stereo 1/8” (3.5mm) TRRS jack.

**Edit**

**Sound Select**

Press and hold the sound select button to preview any kit sound. Try it now by holding sound select and then pressing button 2 to hear the kick 2 drum.

**Write Mode**

Now, let’s place the kick 2 drum into some slots on the sequence. Press the write mode button, and you’ll see the sequence steps lights turn off. This indicates that the currently selected kit sound is not assigned to any steps.
Kick

Let's add the kick drum on the first and third beats of the pattern, as well as on anticipatory final step of the sequence to lead back into the next repeat.

You can think of this 16-step pattern as one 4/4 time measure counted as "one ee and uh two ee and uh three ee and uh four ee and uh"
Fill out the Pattern

You can now fill out the pattern with more drums. To do so, repeat the process of holding sound select and pressing one of the kit sounds. Release the sound select button and then press the sequence steps where you want that sound to play. Note, you must be in write mode to place sounds in step slots, so check that that button is lit up. If not, press it!

Notice how the currently selected sound is lit up blue when you're in write mode so you know which sound you're placing in the sequence step slots.

Snare

Here, I've added snares on the two and four of the count (steps 5 and 13 of the sequence).

Closed Hat

We'll add in closed hi-hats next. Notice how the currently button 5 lights up magenta since it is trying to be both blue to signify the currently selected sound as well as red to signify that it will play on the fifth step of the sequence.
Open Hat

Place an open hat in sequence slot 14.

Clap

Finally, let's use the same method as before to add in a clap sound in sequence slots 2, 8, and 16.
Transport

Play

Time to listen to the beats! For safety, get out of write mode by toggling the write mode button, then, press the play button and the sequence will run and loop.

Record

With a microphone plugged in, such as the inline mic on a set of earbuds, you can record your own samples to any slot in the second bank.
Press and hold the record button, and then hold one of the buttons 9-16 while you sing, talk, scream, beat box, squawk, or otherwise make beautiful sounds. Release both buttons when you’re finished. Each sample can be up to two seconds long.

Now, you can select and sequence any of these sampled-kit sounds just as you did with the built-in kit sounds.

**Beat Repeats and Step Muting**

Once you’ve got a sequence playing, you can have fun altering it during performances by using the beat repeat buttons. Press and hold stutter to create a fast, repeating division of the current beat you’re. The fast stutter works the same, only with an even higher division level.

Jump works a little differently. Hold the jump button and then while it's held, press any one of the sixteen sequence step buttons to instantly jump back to that beat.

A related technique is step muting. Press any of the sequence step buttons to toggle muting for all sounds on that step. It's a fun way to breakdown or build up your pattern.

This video shows some beat repeat effects at work, followed by step muting.

Note some videos were shot with a modified wooden faceplate. You can make your own case mods using the Trellis M4 CAD files found here: https://github.com/adafruit/Adafruit-NeoTrellis-M4-PCB-and-Enclosure

**Tilt Effects**

You can also create interest by adding and tuning the filter and bit crush effects.

The low pass filter (LPF) cuts out higher frequencies (only allowing low frequencies to pass through, hence the name) while the high pass filter (HPF) cuts out lower frequencies.

Press either button and then tilt the Trellis M4 side to side. This changes the cutoff frequency of the filter and sounds totally awesome.

You can also crush the playback sample bit rate with the bit crush effect, and dial it in with the tilting action as well.
Kit Solo

The next grouping are the solo buttons. Pressing either or both of these allows you to cut an entire sound bank in and out during playback, effectively muting either all of the built-in kit sounds or any recorded samples.

Tempo

You can adjust the tempo in increments of two BPM faster or slower by pressing the tempo up or tempo down button. This is great for fine tuning your tempo -- if you want to make bigger changes, press and hold one of the buttons to increase/decrease constantly until you arrive at your tempo destination!

Volume

Tap the volume up or volume down buttons to adjust the output loudness.

Inspiration

There are lots of great resources for 16-step drum patterns, have a look online for some.

Deeper Modification

For advanced synth hackers, if you want to tweak the Arduino code driving this thing, you can head to the GitHub repo to have a look!

FAQ/Troubleshooting

While playing audio back from the NeoTrellis M4 over the TRRS jack, I get an odd 'hissing' sound along with whatever is being played. Anything I should do to address this?

If your NeoTrellis M4 is powered by your computer USB, first thing to try is decoupling any ground loops by using headphones or a powered speaker running from a battery pack.
If it persists, there's a chance you're hearing serial data being transferred over the USB port, which can be noisy. This is the trade off of going with plain DAC output as opposed to I2S. It is meant to be a simple audio playback toy, not audiophile quality :)

You may also want to try powering the NeoTrellis M4 over a USB battery pack instead of your computer.

---

When using the 16-step sequencer, if you press the audio sampler buttons, odd music will play

The 16-step sequencer 'sampler' demo where you can record audio onto the board depends on audio being stored in the flash memory, which is also where circuitpython stores files, so if you play one of the 'prerecorded' audio clips it will play files on the file system!

You can solve this by loading CircuitPython back onto the board, then removing all the WAV files.

---

Why does the NeoTrellis sometimes 'hiccup' and restart after a few seconds?

Some computers, especially Mac's, will write a small amount of data after a few seconds, this causes CircuitPython to (correctly) reboot, and you'll get the audio re-start after a few seconds.

---

After loading the 16-step sequencer, the MIDI arpeggiator or some other UF2 examples, my CIRCUITPY no longer appears!

Correct, these are Arduino programs. Once you're done with these examples, revisit the CircuitPython for NeoTrellis M4 installation page and re-install CircuitPython ()
A few seconds/minutes after playing a sound on the NeoTrellis there's a 'peeeeeeeeeeewwwwwwwwww' sound - what's that?

You're hearing the output capacitors discharge after not playing audio for a while. This normal, not harmful, and occurs with some headphones/speakers.

When I press a button on the NeoTrellis a whole row is activated?

It's possible to accidentally "press" a column of four buttons even when you intend to only press one. This can happen when you press a button at an angle other than straight down on top of the button pad. The elastomer conductive part touches the NeoPixel ground pin against the column which makes the chip think the whole row/column is pressed. It only happens on some buttons and some angles, and is not common.

It's not harmful, and there's a few ways to avoid it:

- Press straight down on buttons not at an angle
- Cut a thicker top panel from wood/plastic/3D printing so that the buttons cannot be pressed at an angle
- Read buttons via the [https://github.com/adafruit/Adafruit_NeoTrellisM4](https://github.com/adafruit/Adafruit_NeoTrellisM4) library which will do a software filtering to remove the duplicate button presses.

Why do sometimes all the buttons get dimmer and the Trellis stops working?

If many of the NeoPixels are at close to full brightness, you will draw too much power from the USB port on your computer. Try lowering the brightness. If you are using the [adafruit_trellism4](https://github.com/adafruit/Adafruit_NeoTrellisM4) library, you can use the `.brightness` setter to lower the brightness (1.0 is max; try 0.2 or 0.3).