



# Neotrellis M4 Live Launcher

Created by Collin Cunningham



<https://learn.adafruit.com/neotrellis-live-launcher>

Last updated on 2023-08-29 03:57:48 PM EDT

# Table of Contents

<b>Overview</b>	<b>3</b>
<ul style="list-style-type: none"><li>• What you'll need</li><li>• Ableton Live &amp; Max for Live</li></ul>	
<b>NeoTrellis Setup</b>	<b>4</b>
<ul style="list-style-type: none"><li>• Assemble the NeoTrellis M4</li><li>• Install the Adafruit NeoTrellis M4 library</li><li>• Create the Live Launcher sketch</li><li>• Upload the code</li></ul>	
<b>Live Setup</b>	<b>9</b>
<ul style="list-style-type: none"><li>• Open a Live set</li><li>• Add the Live Launcher</li><li>• Connect Live to NeoTrellis</li></ul>	
<b>Use it</b>	<b>10</b>

---

# Overview

This project turns your NeoTrellis M4 into a control surface for [Ableton Live](#) () similar to functionality found on a [Novation Launchpad](#) () or [Ableton Push](#) (). It allows you to start and stop 32 individual audio or MIDI clips with the touch of a button and each clip is colored according to values sent from Live's UI.



## What you'll need



[Adafruit NeoTrellis M4 with Enclosure and Buttons Kit Pack](#)

So you've got a cool/witty name for your band, a Soundcloud account, a 3D-printed Daft Punk...

<https://www.adafruit.com/product/4020>



## USB cable - USB A to Micro-B

This here is your standard A to micro-B USB cable, for USB 1.1 or 2.0. Perfect for connecting a PC to your Metro, Feather, Raspberry Pi or other dev-board or...

<https://www.adafruit.com/product/592>

## Ableton Live & Max for Live



This project has only been tested with Ableton Live 10 for MacOS.

Of course, you'll also need an installation of Ableton Live. The Live Launcher plugin device is written in MaxMSP, so you'll need either Live 10 Suite (which includes Max for Live) or Ableton Live 10 + the [Max for Live add-on \(\)](#). The Live Launcher is likely compatible with Live 9, but has not been tested at the time of this writing.

## NeoTrellis Setup

The Live Launcher consists of two pieces of software:

- Neotrellis firmware written in the Arduino IDE
- Live Launcher device plugin for Ableton Live

[Download & install \(\)](#) the Arduino IDE if you haven't already and add support for Adafruit boards via [instructions here \(\)](#).

## Assemble the NeoTrellis M4

Follow the [instructions in this guide \(\)](#) to assemble and test your NeoTrellis M4. Once you've got it up and running, you can move on to programming.

## Install the Adafruit NeoTrellis M4 library



Open Arduino, go to Sketch -> Include Library -> Manage Libraries ... and type NeoTrellis in the search field of the window that appears. Install the Adafruit NeoTrellis M4 Library that appears in the search results.

## Create the Live Launcher sketch

Create a new sketch and delete the default template code which appears within it. Copy the code below, paste it into that new sketch, and save it as Live\_Launcher (or whatever name you see fit).

```
// SPDX-FileCopyrightText: 2018 Collin Cunningham for Adafruit Industries
//
// SPDX-License-Identifier: MIT

/* Live Launcher - Ableton Live controller for Adafruit Neotrellis M4
   by Collin Cunningham for Adafruit Industries
   https://www.adafruit.com/product/3938
*/

#include <Adafruit_NeoTrellisM4.h>

#define WIDTH      8
#define HEIGHT     4
#define N_BUTTONS  WIDTH*HEIGHT
#define NEO_PIN    10
#define NUM_KEYS   32
#define SERIAL_TIMEOUT 1000 //time before giving up on incoming serial data
#define COLOR_DATA_LENGTH 98 //number of total bytes to expect in incoming color
data
//4 bytes per color * 32 buttons = 128 bytes + 2 header bytes = 130 bytes total
#define BUTTON_DATA_LENGTH 5 //clip playing status message for each button
#define PULSE_DURATION 350 //length of 'now playing' pulse

unsigned long readTime; //time we start reading serial buffer
```

```

unsigned long lastPulseTime;
bool pulseOn = false;
uint8_t packetbuffer[COLOR_DATA_LENGTH]; //store incoming serial data
uint8_t colors[96];

Adafruit_NeoTrellisM4 trellis = Adafruit_NeoTrellisM4();

const byte ROWS = HEIGHT; // four rows
const byte COLS = WIDTH; // eight columns
byte trellisKeys[ROWS][COLS] = { //define the symbols on the buttons of the keypads
  {1, 2, 3, 4, 5, 6, 7, 8},
  {9, 10, 11, 12, 13, 14, 15, 16},
  {17, 18, 19, 20, 21, 22, 23, 24},
  {25, 26, 27, 28, 29, 30, 31, 32}
};
byte rowPins[ROWS] = {14, 15, 16, 17}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {2, 3, 4, 5, 6, 7, 8, 9}; //connect to the column pinouts of the keypad
Adafruit_Keypad customKeypad = Adafruit_Keypad( makeKeymap(trellisKeys), rowPins, colPins, ROWS, COLS); //initialize keypad
extern const uint8_t gamma8[];

boolean pressed[N_BUTTONS] = {false}; // Pressed state for each button
uint8_t playing[N_BUTTONS] = {0}; // Playing state for each button

void setup() {
  Serial.begin(57600);
  // while (!Serial) {}

  trellis.begin();
  trellis.setBrightness(255);
  trellis.fill(0);
}

void loop() {
  unsigned long startTime = millis();

  if (Serial.available() >= BUTTON_DATA_LENGTH) {
    parseData();
  }

  //send press events to Live via serial
  trellis.tick();
  while (trellis.available()) {
    keypadEvent e = trellis.read();
    uint8_t i = e.bit.KEY;
    if (e.bit.EVENT == KEY_JUST_PRESSED) {
      pressed[i] = true;
      Serial.write(i);
    }
    else if (e.bit.EVENT == KEY_JUST_RELEASED) {
      pressed[i] = false;
    }
  }

  if ((startTime - lastPulseTime) >= PULSE_DURATION) {
    //flash any clip which is playing
    if (pulseOn) {
      for (uint8_t i = 0; i < N_BUTTONS; i++) {
        setPixelWithColorsIndex(i, false);
      }
    }
    else {
      for (uint8_t i = 0; i < N_BUTTONS; i++) {
        if (playing[i]) {
          setPixelWithColorsIndex(i, true);
        }
      }
    }
  }
}

```

```

        else setPixelWithColorsIndex(i, false);
    }
}
pulseOn = !pulseOn;
lastPulseTime = millis();
}
}

void parseData() {
    //check for ! start char
    if (Serial.read() == '!') {
        uint8_t id = Serial.read();

        //check for Color data
        if (id == 'C') {
            for (int i = 0; i < 96; i++) {
                colors[i] = Serial.read();
            }

            //color button leds
            for (int i = 0; i < N_BUTTONS; i++) {
                setPixelWithColorsIndex(i, false);
            }
        }

        //check for clip status data
        else if (id == 'B') {
            uint8_t x = Serial.read();
            uint8_t state = Serial.read();
            uint8_t y = Serial.read();

            if (state == 0) { //if state is 0, all clips in track are stopped
                for (int i = 0; i < HEIGHT; i++) {
                    uint8_t index = i * WIDTH + x;
                    playing[index] = state;
                }
            }

            else { //all other clips in track should stop
                for (int i = 0; i < HEIGHT; i++) {
                    uint8_t index = i * WIDTH + x;
                    playing[index] = 0;
                }
                //save playing state
                uint8_t index = y * WIDTH + x;
                playing[index] = state;
            }
        }
    }
}

void setPixelWithColorsIndex(int i, bool dimmed) {

    uint8_t red = colors[i * 3];
    uint8_t green = colors[i * 3 + 1];
    uint8_t blue = colors[i * 3 + 2];
    if (dimmed) {
        setPixelWithGamma(i, red / 2, green / 2, blue / 2);
    }
    else {
        setPixelWithGamma(i, red, green, blue);
    }
}

void setPixelWithGamma(int pixelNumber, uint8_t red, uint8_t green, uint8_t blue) {
    uint32_t color = trellis.Color(

```

```

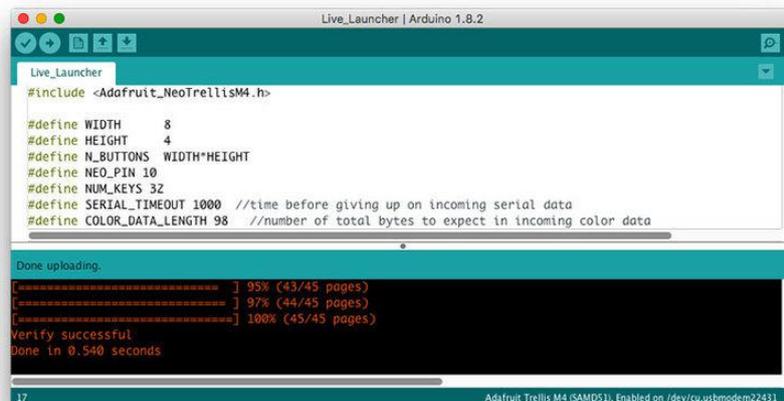
        pgm_read_byte(&gamma8[red]),
        pgm_read_byte(&gamma8[green]),
        pgm_read_byte(&gamma8[blue]));

    trellis.setPixelColor(pixelNumber, color);
}

const uint8_t PROGMEM gamma8[] = {
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
    0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,
    1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2,
    2, 3, 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 5, 5, 5,
    5, 6, 6, 6, 6, 7, 7, 7, 7, 8, 8, 8, 9, 9, 9, 10,
    10, 10, 11, 11, 11, 12, 12, 13, 13, 13, 14, 14, 15, 15, 16, 16,
    17, 17, 18, 18, 19, 19, 20, 20, 21, 21, 22, 22, 23, 24, 24, 25,
    25, 26, 27, 27, 28, 29, 29, 30, 31, 32, 32, 33, 34, 35, 35, 36,
    37, 38, 39, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 50,
    51, 52, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 66, 67, 68,
    69, 70, 72, 73, 74, 75, 77, 78, 79, 81, 82, 83, 85, 86, 87, 89,
    90, 92, 93, 95, 96, 98, 99, 101, 102, 104, 105, 107, 109, 110, 112, 114,
    115, 117, 119, 120, 122, 124, 126, 127, 129, 131, 133, 135, 137, 138, 140, 142,
    144, 146, 148, 150, 152, 154, 156, 158, 160, 162, 164, 167, 169, 171, 173, 175,
    177, 180, 182, 184, 186, 189, 191, 193, 196, 198, 200, 203, 205, 208, 210, 213,
    215, 218, 220, 223, 225, 228, 231, 233, 236, 239, 241, 244, 247, 249, 252, 255
};

```

## Upload the code



Now, just a few more steps in the Arduino IDE and you'll have the NeoTrellis ready for communication with Ableton Live:

1. Reset your NeoTrellis by double-clicking the recessed reset button on the back. NeoTrellis is ready to be programmed when the rear LED turns green.
2. Go to Tools -> Board and select Adafruit Trellis M4 (SAM51) from the list.
3. Go to Tools -> Port and select the port which name contains Adafruit Trellis M4 in the name.
4. Upload the sketch by clicking Sketch -> Upload.

---

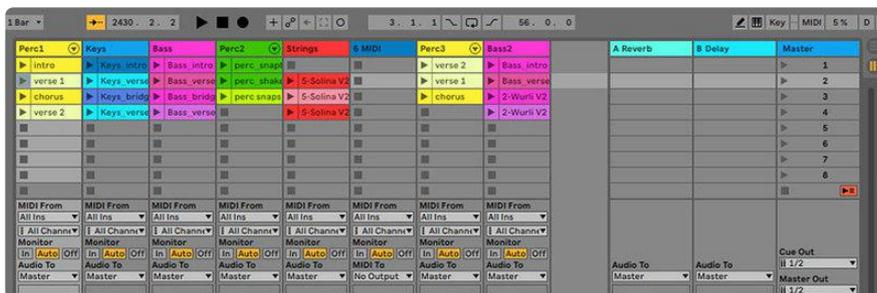
# Live Setup

Download the Live Launcher device for Ableton Live by clicking the button below:



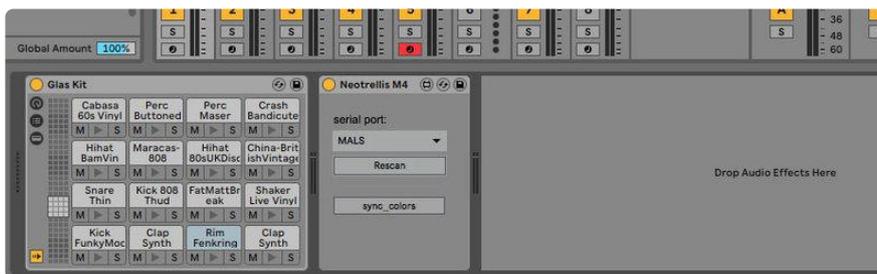
Unzip the downloaded file and store the resulting Neotrellis\_M4.amxd file somewhere safe.

## Open a Live set



In Ableton Live, create a new Live set and add some audio and/or MIDI clips to the first four slots of tracks 1-8 in Session view - or use an existing set that has a variety of clips loaded into Session view.

## Add the Live Launcher



Locate the Neotrellis\_M4.amxd file in the folder you saved it, drag it into the Live window, and drop it onto any track - just be sure to only add one instance of the plugin device.

## Connect Live to NeoTrellis

Ensure your NeoTrellis is connected to your computer via USB and click the rescan button in the NeoTrellis M4 Live device to update the list of available serial devices.



Click the serial port pulldown menu to see a list of available serial devices, and choose the serial port that corresponds to your NeoTrellis M4.

As seen above, my NeoTrellis appeared as usbmodemMID1. If your NeoTrellis does not appear, click the rescan button and trying again.

Click the sync\_colors button to send clip color data over to your NeoTrellis. Once you see your NeoTrellis light up with colors corresponding to tracks 1 - 8 in Live, you're ready to roll.

---

## Use it

The Live Launcher allows you to control the top four clip slots of tracks 1 through 8. Usage is simple:

- Press a Neotrellis button to toggle the playing state of a corresponding Live clip
- Neotrellis buttons will flash to indicate when a clip is playing
- Click the sync\_colors button after moving or recoloring clips in Live

Keep in mind that a clip may take some time to start playing after being triggered - this is determined by each clip's launch settings & tempo in Live.

Here's a quick demo, showing basic control with a simple track layout:

Enjoy making music.