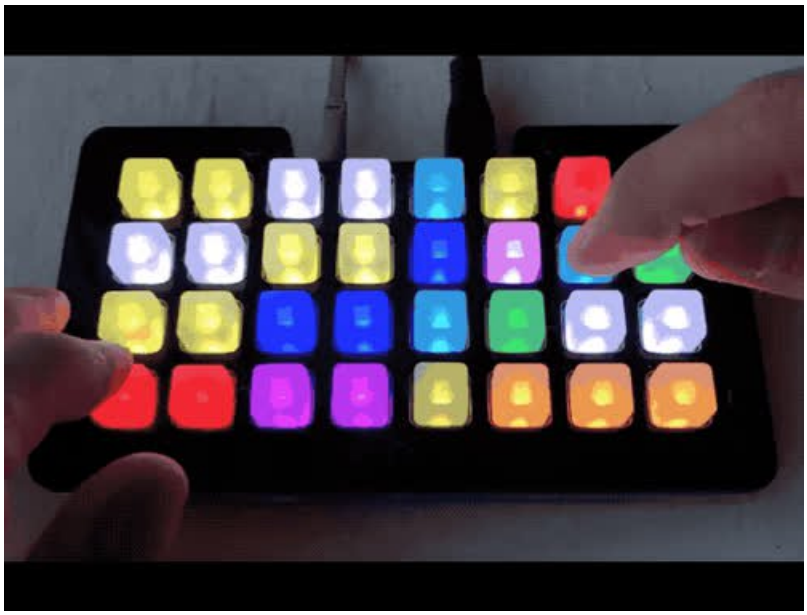
adafruit learning system

Star Trek Soundboard with NeoTrellis

Created by Dano Wall



Last updated on 2018-11-22 02:00:43 AM UTC

Guide Contents

Guide Contents 2
Overview 3
About the NeoTrellis M4 3
What to Buy 3
Prepare Audio Files 5
No Mix & Match! 5
Soundboard Code 6
Navigating the NeoTrellis 6
Add Audio 6
Soundboard Code 7
Boot up: Color! 11
Making Changes 11
Troubleshooting 12
Use It! 14
Exploring Further 15
Other Sound Sources 15
Recording Your Own Custom Sounds 15
Custom Colors 16

Overview

This guide shows you how to build your own soundboard (https://adafru.it/C-V). A soundboard is a set of buttons that will play sounds or music clips when the corresponding button is pressed.

This project uses the NeoTrellis M4 (https://adafru.it/CZe) to create a soundboard capable of playing any 32 audio clips, and puts on a good light show at the same time too!

Soundboards are commonly used for sound effects or to select various pieces of dialog on command. They are sometimes used for pranks (as in this famous example (https://adafru.it/D1G)), or by DJs to interject prerecorded sounds.

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, 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.

What to Buy

1 x NeoTrellis M4 with Enclosure and Buttons Pack

NeoTrellis mainboard with silicone elastomer 4x4 pads and acrylic enclosure included.

OUT OF STOCK

1 x Micro USB cable

Standard A to micro-B USB cable - 3ft

ADD TO CART

1 x USB Powered Speakers

Add some extra boom to your audio project with these powered loudspeakers.

ADD TO CART

1 x 3.5mm Male/Male Stereo Cable

Seamlessly transmit high-quality stereo audio with this 3.5mm (otherwise known as "1/8 inch") stereo cable.

ADD TO CART

Prepare Audio Files

This soundboard requires 32 different audio clips. In this project we will use clips from Star Trek (the original late 1960s TV series) to demonstrate how to work with .mp3 files, resulting in a nerd-tastic tool.

Star Trek sounds can be purchased online here (https://adafru.it/C-J) and/or here (https://adafru.it/C-K).

Other audio clips can also be downloaded from sites the specialize in Star Trek fandom (https://adafru.it/D1H) or celebrity impersonations (https://adafru.it/D1I).

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 NeoTrellis M4.

See this guide on how to convert audio files (https://adafru.it/CQM).

No Mix & Match!

Make sure your audio files are exported as **16-bit PCM WAV** at **22,050 Hz** and they are **all Stereo** or **all Mono** *-no mix and match*!

Soundboard Code

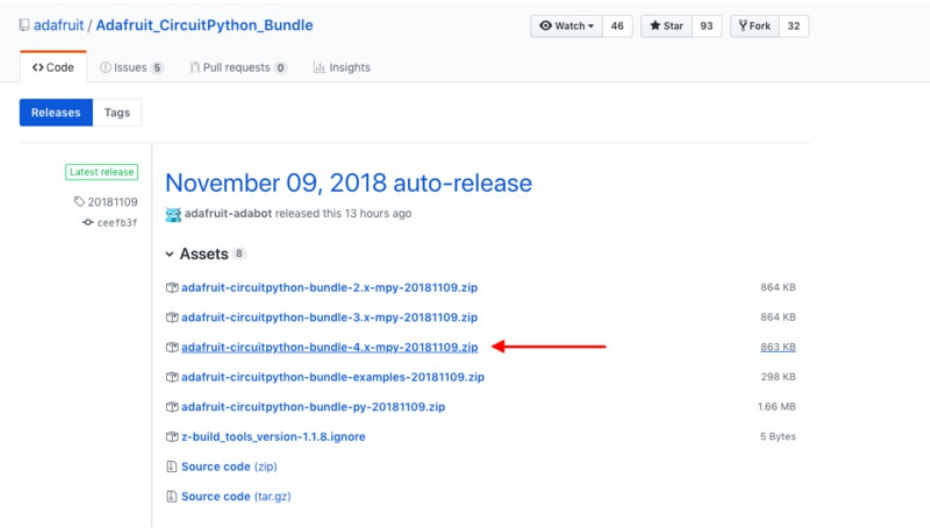
Navigating the NeoTrellis

To get your NeoTrellis M4 set up to run the soundboard code, follow these steps:

1) Don't forget to update the bootloader for NeoTrellis (https://adafru.it/C-N)from the NeoTrellis M4 guide

2) Install the latest CircuitPython for NeoTrellis (https://adafru.it/C-O)from the NeoTrellis M4 guide

3) Get the latest 4.0 library pack (https://adafru.it/zB-), unzip it, and drag the libraries you need over into the **/lib** folder on **CIRCUITPY**. The latest library package includes support for NeoTrellis.
https://github.com/adafruit/Adafruit_CircuitPython_Bundle/releases/ (https://adafru.it/zB-)



For this project you will need the following libraries:

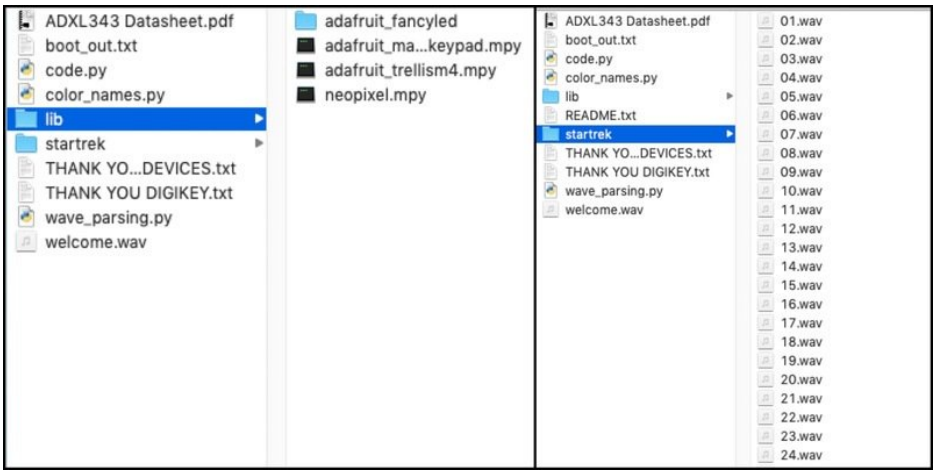
- **adafruit_trellism4.mpy**
- **adafruit_fancyled** folder
- **neopixel.mpy**
- **adafruit_matrixkeypad.mpy**

Add Audio

Ensure your created sound clips are numbered 01.wav to 32.wav (don't forget those leading zeroes!). They all must be 16 Bit, 22KHz WAV files in mono (single channel audio) or stereo (2 channel audio) and cannot be mixed mono & stereo.

Connect your NeoTrellis to your computer via a known good USB cable. Your operating system will show a new "thumb drive" named **CIRCUITPY** is available. If you do not see this happening, visit the NeoTrellis M4 introductory tutorial (https://adafru.it/D1J) to ensure CircuitPython has been loaded onto the device.

Next, in your file explorer/Finder, create a folder on the **CIRCUITPY** flash drive labeled `/startrek`. You can then drag and drop your audio files into that directory (ensuring the files are named from 01.wav through 32.wav). You can change the name of the folder later in the code if you want.



Soundboard Code

Onto the final step, the code itself!

Copy `code.py` from the link below and put it in **CIRCUITPY** root directory. You can work with this code in any text editing application, or open and save with Mu (https://adafru.it/ANO) if you prefer.

Also copy the linked `color_names.py` file into **CIRCUITPY** root directory.

```
import time
import board
import audioio
import adafruit_fancyled.adafruit_fancyled as fancy
import adafruit_trellism4
from color_names import * # pylint: disable=wildcard-import,unused-wildcard-import


PLAY_SAMPLES_ON_START = False

SAMPLE_FOLDER = "/startrek/"  # the name of the folder containing the samples
# This soundboard can select up to *32* sound clips! each one has a filename
# which will be inside the SAMPLE_FOLDER above, and a *color* in a tuple ()
SAMPLES = [("01.wav", RED),
           ("02.wav", ORANGE),
           ("03.wav", YELLOW),
           ("04.wav", GREEN),
           ("05.wav", TEAL),
           ("06.wav", BLUE),
           ("07.wav", PURPLE),
           ("08.wav", PINK),
           ("09.wav", RED),
           ("10.wav", ORANGE),
           ("11.wav", YELLOW),
           ("12.wav", GREEN),
           ("13.wav", TEAL),
           ("14.wav", BLUE),
           ("15.wav", PURPLE),
           ("16.wav", PINK),
           ("17.wav", RED),
           ("18.wav", ORANGE),
           ("19.wav", YELLOW),
           ("20.wav", GREEN)
```

```
            ("20.wav", GREEN),
            ("21.wav", TEAL),
            ("22.wav", BLUE),
            ("23.wav", PURPLE),
            ("24.wav", PINK),
            ("25.wav", RED),
            ("26.wav", ORANGE),
            ("27.wav", YELLOW),
            ("28.wav", GREEN),
            ("29.wav", TEAL),
            ("30.wav", BLUE),
            ("31.wav", PURPLE),
            ("32.wav", PINK)]

# For the intro, pick any number of colors to make a fancy gradient!
INTRO_SWIRL = [RED, GREEN, BLUE]
# the color for the selected sample
SELECTED_COLOR = WHITE

# Our keypad + neopixel driver
trellis = adafruit_trellism4.TrellisM4Express(rotation=0)

# 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 = audioio.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)
    except OSError:
        # no biggie, they probably deleted it
        pass

# Parse the first file to figure out what format its in
channel_count = None
bits_per_sample = None
sample_rate = None
with open(SAMPLE_FOLDER+SAMPLES[0][0], "rb") as f:
    wav = audioio.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!")

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

# turn on maybe play all of the buttons
for i, v in enumerate(SAMPLES):
    filename = SAMPLE_FOLDER+v[0]
    try:
        with open(filename, "rb") as f:
            wav = audioio.WaveFile(f)
            print(filename,
                  "%d channels, %d bits per sample, %d Hz sample rate " %
                  (wav.channel_count, wav.bits_per_sample, wav.sample_rate))
            if wav.channel_count != channel_count:
                pass
            if wav.bits_per_sample != bits_per_sample:
                pass
            if wav.sample_rate != sample_rate:
                pass
            trellis.pixels[(i%8, i//8)] = v[1]
            if PLAY_SAMPLES_ON_START:
                audio.play(wav)
                while audio.playing:
                    pass
    except OSError:
        # File not found! skip to next
        pass

def stop_playing_sample(playback_details):
    print("playing: ", playback_details)
    audio.stop()
    trellis.pixels[playback_details['neopixel_location']] = playback_details['neopixel_color']
    playback_details['file'].close()
    playback_details['voice'] = None

current_press = set()
currently_playing = {'voice' : None}
last_samplenum = None
while True:
    pressed = set(trellis.pressed_keys)
    #if pressed:
    #    print("Pressed:", pressed)

    just_pressed = pressed - current_press
    just_released = current_press - pressed

    #if just_pressed:
    #    print("Just pressed", just_pressed)
    for down in just_pressed:
        sample_num = down[1]*8 + down[0]
        print(sample_num)
        try:
            filename = SAMPLE_FOLDER+SAMPLES[sample_num][0]
            f = open(filename, "rb")
            wav = audioio.WaveFile(f)

            # is something else playing? interrupt it!
```

```
            # is something else playing? interrupt it!
            if currently_playing['voice'] != None:
                print("Interrupt")
                stop_playing_sample(currently_playing)

            trellis.pixels[down] = WHITE
            audio.play(wav)
            # voice, neopixel tuple, color, and sample, file handle
            currently_playing = {
                'voice': 0,
                'neopixel_location': down,
                'neopixel_color': SAMPLES[sample_num][1],
                'sample_num': sample_num,
                'file': f}
        except OSError:
            pass # File not found! skip to next

    #if just_released:
    #    print("Just released:", just_released)

    # check if any samples are done
    if not audio.playing and currently_playing['voice'] != None:
        stop_playing_sample(currently_playing)

    time.sleep(0.01)  # a little delay here helps avoid debounce annoyances
    current_press = pressed
```

```
RED = 0xFF0000
MAROON = 0x800000
ORANGE = 0xFF8000
YELLOW = 0xFFFF00
OLIVE = 0x808000
GREEN = 0x008000
AQUA = 0x00FFFF
TEAL = 0x008080
BLUE = 0x0000FF
NAVY = 0x000080
PURPLE = 0x800080
PINK = 0xFF0080
WHITE = 0xFFFFFF
BLACK = 0x000000
```

Your **CIRCUITPY** flash drive should now include the following files:

1. `code.py` main program
2. `color_names.py` color definition file
3. `/lib` directory containing the necessary CircuitPython 4.x libraries
4. `/startrek` directory with 32 wav file audio clips names 01.wav, 02.wav, . . ., 32.wav

Boot up: Color!

Once you have all your files set up, your board should reboot and start running your code automatically.

The boot-up sequence illuminates all neopixels in a rainbow pattern, then goes dark for a second before coming back on in an array of Star Trek-inspired colors.

Making Changes

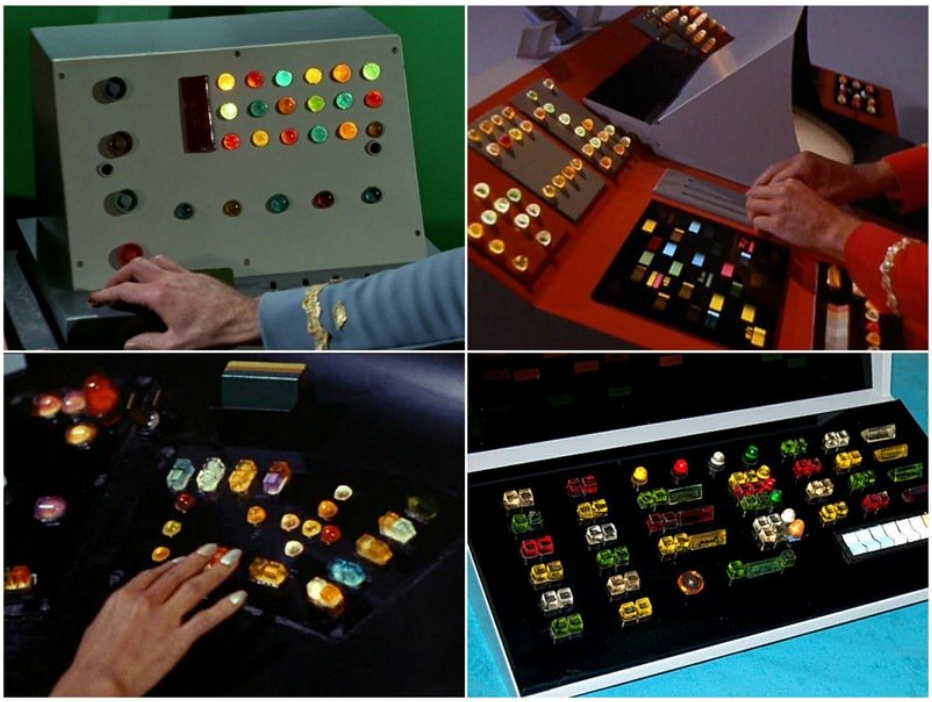
If you'd like to change the code, you can do so as follows:

The line `INTRO_SWIRL = [PINK, TEAL, YELLOW]` can be changed to be any three color names listed in `color_names.py`. You can even put more color values in that file and refer to them in your code.

In the definition of `SAMPLES` near the top of the code - it lists a file name and the color of the button. Feel free to

color the buttons how you wish using the color names in `color_names.py`.

While Star Trek used lots of primary colors (as an early color TV program), your other soundboards may be different. Say, different shades of green for Teenage Mutant Turtles, etc.



Troubleshooting

You can use the serial capability in Mu (https://adafru.it/ANO) or a serial terminal program to connect to the NeoTrellis M4 to interact with the CircuitPython prompt. You can see error messages, restart the program, etc. If you use a terminal program, set it to the COM port that appears when the device is plugged in. In Windows, Device Manager will show a keyboard when NeoTrellis is plugged in. Right click, Properties, Hardware will show the COM port. Connect at 9600 baud.

Problem: I don't see the **CIRCUITPY** drive when I plug the NeoTrellis M4 into my computer

Solution: Ensure CircuitPython is installed via this guide (https://adafru.it/D1J).

Problem: The code does not run.

Solution: Check the files on the **CIRCUITPY** drive. /lib should have the latest CircuitPython 4.x libraries, /startrek holds the 32 sound files, `code.py` and `color_names.py` should be in the main (root) directory.

Problem: Not enough disk space for all the files.

Solution: You cannot use long stereo clips or songs for each button, there is not enough flash memory space. You can reduce the number of files below 32. You can trim your clips to make them shorter. Or you can ensure all your clips are mono instead of stereo. Advanced users can remove libraries in **/lib** for sensors that are not used by your device to

free up small amounts of flash but please do not delete any libraries used by the program or low level input/output libraries.

Problem: My board isn't booting up!

Solution: Make sure your **/lib** folder is set up with the libraries in the 4.0 latest release and includes trellism4.mpy in its contents.

Problem: I'm not hearing any sounds!

Solution: Check that all your files are formatted as Stereo 16-bit, 22,050Hz, PCM

Use It!

Your soundboard is now ready to make some noise! Have fun making music, or interjecting sound effects into conversation.

Some clips are noticeably longer than others, and you will notice that one button press is able to interrupt the audio clip that came before it, so you can press as messily as you like and the soundboard will keep up with you.

Exploring Further

Other Sound Sources

You can use any 32 sounds you want, so you are not limited to Star Trek. If you search the web for "Soundboard", you can find several sites that host groups of sound files.

You will most likely have to convert the files you do find to the WAV format as noted in the Prepare Audio Files section.

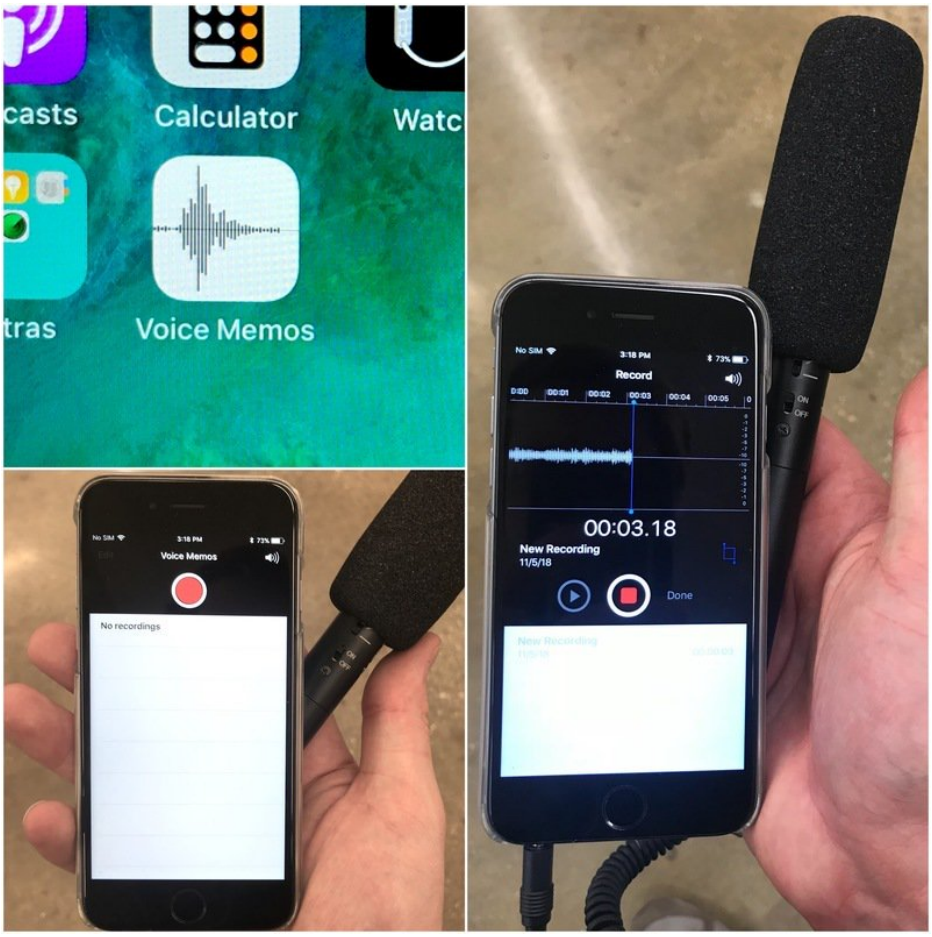
Some sounds may be Open Source (or Creative Commons unrestricted CC 0) and others may be commercial. If you want to use your soundboard in a commercial way or to record on Youtube, it is best you either use unrestricted sounds or buy a license for the material you use. This is the reason in this tutorial we state the Star Trek sounds may be purchased via CD or download.

Adafruit is collecting unrestricted sounds on GitHub (https://adafru.it/D0z) for Trellis use. There are also sites with unrestricted samples such as freesound.org (https://adafru.it/D0A) you may wish to look through.

Recording Your Own Custom Sounds

If you'd like to create your own custom soundboard, you can use your phone to record audio clips and load those onto your NeoTrellis!

Any recording app will work to gather sound clips, and a portable microphone is nice as well (though not necessary!)



Custom Colors

You can change the soundboard to have any array of custom colors you want.

Do this by editing the `color_names.py` file to update or add colors, then updating the `code.py` file to assign colors to any of the 32 buttons in the array.