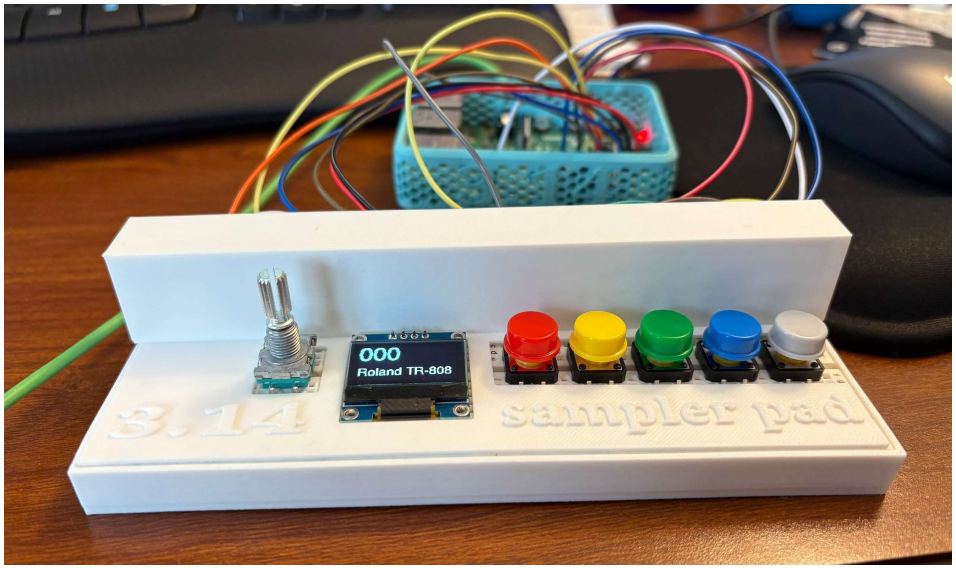


Sampler Pad

Alexander Kniss & Matthieu Banks

Introduction

An attempt to combine both artistic expression and tech, Alex and I created a sampler with 5 buttons that allow you to combine different sounds in inventive ways.

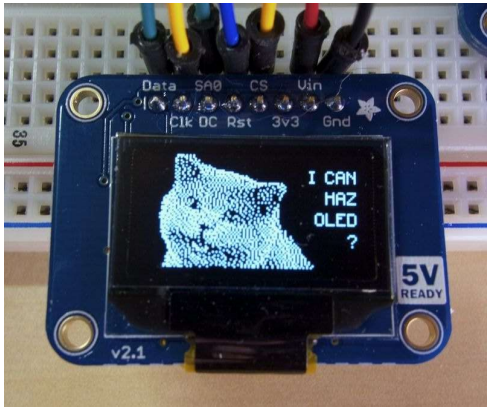
Also created as an accessible alternative to expensive instruments.

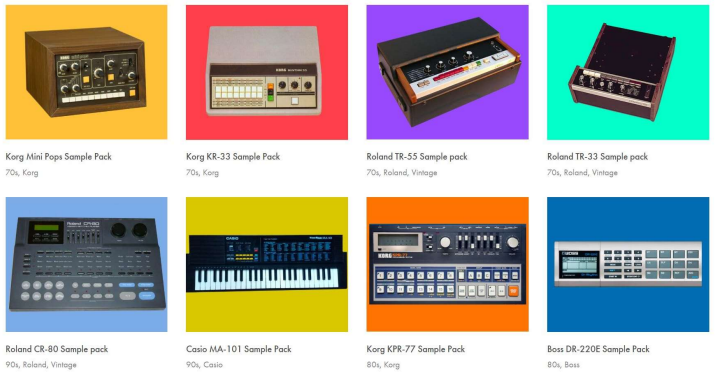
Project Objectives/features

- Easily Accessible Sounds for music making
- Variation in sound packs
- Ability to mix and match different sounds with each other

Hardware and Software Components

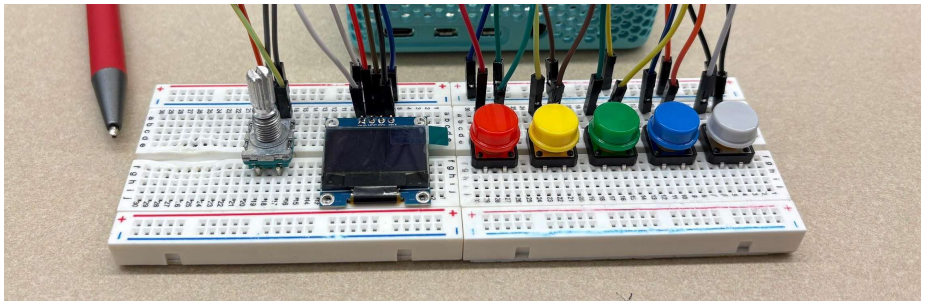
- Adafruit tactile buttons
  - Larger and easier to press
- Rotary encoder
  - Used as a dial to select different sound packs
- OLED display
  - Display for sound pack number and name
- Drum Sample Packs from drumkito.com
  - Sounds from authentic drum machines

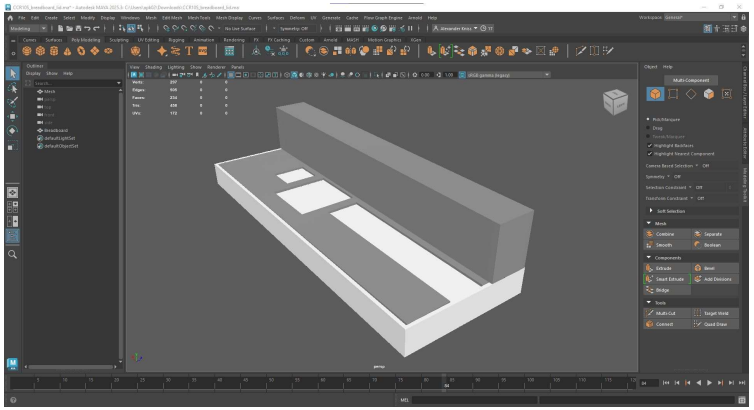




Project Approach

- The first thing we did was to come up with a design for our sampler pad. The design we ended up with utilized 2 breadboards, 5 buttons in a row on the right side, a rotary encoder on the left side and an OLED display in the middle.
- We also needed a container for a sampler pad. The bottom case is from printables.com, and we designed the top lid in Blender and Maya. Using our best measurements, we mapped out holes for our components, and built part of the lid higher to hide the wires underneath.

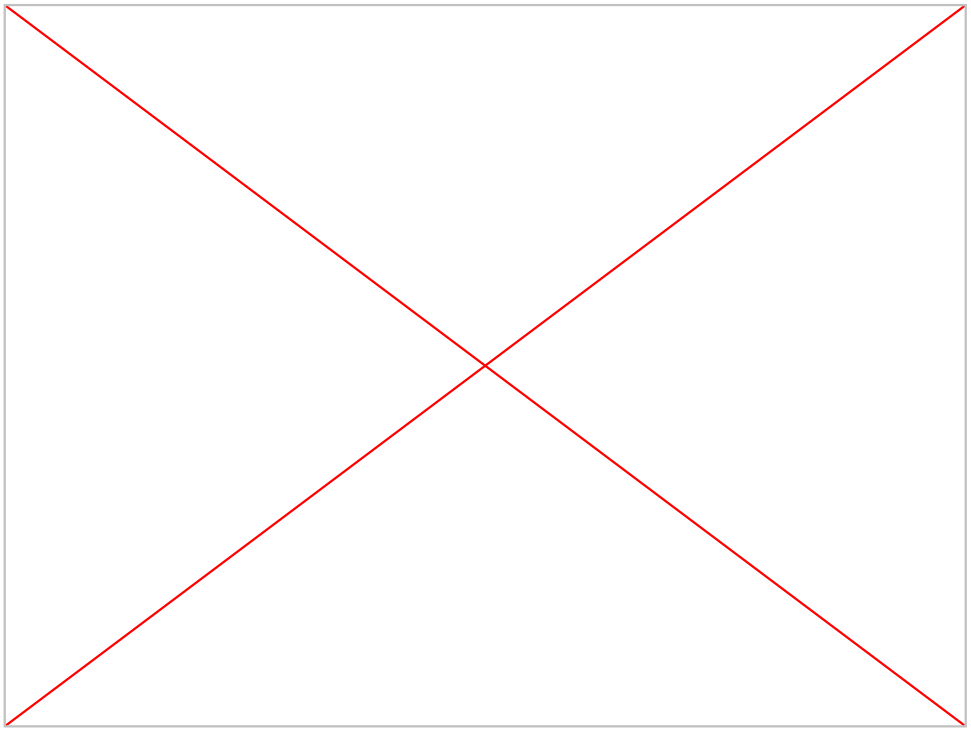




Project Approach Pt. 2

- The software portion started with implementing basic code using previous exercises as reference. We tested to make sure the buttons produced output, the OLED displayed text, and the rotary encoder changed the current pack number correctly.
- To store the different sounds, we used lists that referred to specific sound files from each pack to be used by each specific button. When a button is pressed, it would get the sound file path from the list based on the current pack number, and play the sound using pygame.

Demonstration



Conclusion

There were a few different difficulties encountered during the creation of our project, such as

- Getting the lid to precise measurements
  - required 2 prints and printers in IRL being in use
- Coding the rotary encoder to work properly
  - ended up using a different implementation than in exercise 13.7
- Problems with sound delay
  - experimented with different python modules to play sound
  - pygame was not updated to the latest version

Conclusion

Overall, our sampler pad turned out well. Some further improvements would be

- Expanding the sound bank
  - Add sound effects and loops in addition to simple drum sounds
- Working to eliminate any delay between button presses and sound output
  - May require a more advanced python module
- Adding a volume knob
  - pygame allows for channel volume to be adjusted
- Making buttons programmable by user
  - Create custom mappings using only the pad itself, would require an addition button

Python Code

```
from gpiozero import Button
import pygame

import board
from PIL import Image, ImageDraw, ImageFont
import adafruit_ssd1306
from time import sleep

# Set up display
i2c = board.I2C()
disp = adafruit_ssd1306.SSD1306_I2C(128, 64, i2c, addr=0x3C)
small_font = ImageFont.truetype('FreeSans.ttf', 18)
large_font = ImageFont.truetype('FreeSans.ttf', 33)
disp.fill(0)
disp.show()

# Make an image to draw on in 1-bit color.
width = disp.width
height = disp.height
image = Image.new('1', (width, height))
draw = ImageDraw.Draw(image)

# Display a message on 3 lines, first line big font
def display_message(top_line, line_2):
    draw.rectangle((0,0,width,height), outline=0, fill=0)
    draw.text((0, 0),  top_line, font=large_font, fill=255)
    draw.text((0, 40),  line_2, font=small_font, fill=255)
    disp.image(image)
    disp.show()

pygame.mixer.init()

red = Button(18)
yellow = Button(23)
green = Button(24)
blue = Button(25)
white = Button(21)
turn_left = Button(27)
turn_right = Button(17)
```

Assigning buttons to gpio pins

Taken from 15.6 with modification to text size

Using lists as data storage

```
pack = 0
pack_names = ["Roland TR-808", "Yamaha MR-10", "Boss DR-220A"]

root = "/home/akniss/Downloads/"

bass = [pygame.mixer.Sound(root+"Roland TR-808/BD/BD0000.WAV"),
        pygame.mixer.Sound(root+"mr10/kick1.wav"),
        pygame.mixer.Sound(root+"Boss_DR-220A/DR220Kick.wav")]

snare = [pygame.mixer.Sound(root+"Roland TR-808/SD/SD0010.WAV"),
         pygame.mixer.Sound(root+"mr10/snare.wav"),
         pygame.mixer.Sound(root+"Boss_DR-220A/DR220Snare.wav")]

closed = [pygame.mixer.Sound(root+"Roland TR-808/CH/CH.WAV"),
          pygame.mixer.Sound(root+"mr10/chihat.wav"),
          pygame.mixer.Sound(root+"Boss_DR-220A/DR220Hat_C.wav")]

openhi = [pygame.mixer.Sound(root+"Roland TR-808/OH/OH75.WAV"),
          pygame.mixer.Sound(root+"mr10/ohihat.wav"),
          pygame.mixer.Sound(root+"Boss_DR-220A/DR220Hat_O.wav")]

crash = [pygame.mixer.Sound(root+"Roland TR-808/CY/CY0075.WAV"),
         pygame.mixer.Sound(root+"mr10/cymbal.wav"),
         pygame.mixer.Sound(root+"Boss_DR-220A/DR220Crash.wav")]

bass_channel = pygame.mixer.Channel(0)
snare_channel = pygame.mixer.Channel(1)
closed_channel = pygame.mixer.Channel(2)
open_channel = pygame.mixer.Channel(3)
crash_channel = pygame.mixer.Channel(4)
```

Assigning channels so that more than one sound can be played at once

Python Code

```
def get_red():
    bass_channel.play(bass[pack])

def get_yellow():
    snare_channel.play(snare[pack])

def get_green():
    closed_channel.play(closed[pack])

def get_blue():
    open_channel.play(openhi[pack])

def get_white():
    crash_channel.play(crash[pack])
```

Plays sounds based on the buttons pressed and current sound pack

```
def display():
    global pack
    pack_number = '{:03}'.format(pack)
    pack_message = pack_names[pack]
    display_message(pack_number, pack_message)
```

Method to display pack number and name onto the OLED

```
def increase():
    global pack
    if pack < len(pack_names):
        pack += 1
        display()
    else:
        sleep(0.1)

def decrease():
    global pack
    if pack > 0:
        pack -= 1
        display()
    else:
        sleep(0.1)
```

Changes the global pack variable and displays onto the OLED, but doesn't allow to go over number of packs

Initial display

```
display()
```

```
red.when_pressed = get_red
yellow.when_pressed = get_yellow
green.when_pressed = get_green
blue.when_pressed = get_blue
white.when_pressed = get_white
turn_left.when_pressed = decrease
turn_right.when_pressed = increase
```

Mapping button presses to the appropriate function