

MA Commercial Music Producer

-Tileyard Education, London,

2021-2022

Thomas Benjamin Marlow

Student ID: 2005496

Development Module- Element D2b- Essay

Module Code: ADAD7005

Word Count: 4000 +- 10% (4267)

Introduction

This essay will detail the research and approach taken to create my instrument the 'Box Of Chops' in response to the given task of producing a physical electronic instrument. A brief overview of the product will be discussed, with detailed explanations regarding component choices clarified later on in the essay. The project outline will be presented, followed by details of the prototyping phases covering research, construction and coding. The thought process behind the visual branding will be examined, followed by a comparison with existing commercial products. Finally, the findings from user feedback will be analysed, and a reflection on the product and future developments will be discussed.

Instrument Overview

To briefly summarise, the 'Box Of Chops' is a unique handheld vocal sampler which has an ergonomic design and tactile feel reminiscent of a handheld gaming device. Designed and based around an Arduino MKR Zero microcontroller (Arduino MKR ZERO (I2S bus & SD for sound, music & digital audio data), 2022) each switch is connected to an input on the board and the chosen output pins follow the I2S standard, connecting to a stereo DAC breakout board (Adafruit I2S Stereo Decoder - UDA1334A Breakout - Pimoroni, 2022).

To play samples, a Micro SD card loaded with 16 bit 44.1kHz WAV files is inserted into the slot on the Arduino MKR Zero board. The 'Box Of Chops' comes pre-loaded with 12 vocal samples and 8 drum samples. The 12 vocal samples are duplicates of each other pitched one semitone apart, forming the 12-note chromatic scale. Users could upload their own samples to the SD card, however a small change to the code would be necessary to make the board read the newly named files correctly. Optionally, a new naming system could be created (e.g. 'SAMPLE1.WAV', 'SAMPLE2.WAV') which if adhered to, would mean any samples uploaded to the SD card named in this manner would be read and played automatically.

The stereo I2S decoder board produces a professional quality of audio through a 16 bit 44.1kHz stereo line level output. This can be connected directly to the line input of an interface without impedance issues.

The device is powered by a 150mAh LiPo battery (LiPo Battery Pack - Pimoroni, 2022) which can be turned on via the rocker switch on the back of the unit and charged via the 5V micro USB port.

To play the stored samples, the 'Chop It' button- a capacitive touch sensor (Adafruit Standalone Momentary Capacitive Touch Sensor Breakout - Pimoroni, 2022) will play the sample for as long as it is held, looping if the sample reaches its end. Once released, the sample will stop playing. When pressed again, the sample will start playing from the beginning.

Using the 16 position BCD (Binary coded decimal) rotary switch (BCKS1002RS | Lorlin, 16 Position, BCD Rotary Switch, 150 mA @ 250 V ac, Through Hole | RS Components, 2022) the user can select a key to match their track. Once selected, the user chooses whether their track is major or minor using the slide switch. Depending on the position of the switch, the code decides exactly which 7 of the 12 notes fit the key and tonality and maps them to the corresponding joystick positions. Having chosen the key and tonality, the user can be certain that any sounds produced from the device will be in key and will work with their track.

To select which of the 7 samples (aka the 7 notes of a key) to play back when the touch sensor is pressed, an 8-position joystick (RKJXL100401V | Alps Alpine Center-Push, 8-Axis Joystick Switch Lever, Momentary | RS Components, 2022) is implemented. The 7 samples which relate to the selected key and tonality are mapped to the joystick positions in an anti-clockwise manner, as can be seen in Appendix A. The spare 8th position of the joystick is programmed to randomly select one note from within the key. This adds an element of randomization and could inspire new ideas.

In addition to being a vocal sampler, the device uses one of the four spare positions of the rotary encoder to act as a creative drum sampler. 8 drums samples loaded onto the SD card are mapped to each of the 8 joystick positions; and by tapping the touch sensor to a beat whilst moving the joystick, the user can play a drum pattern in a unique and creative way.

Project outline: Defining the concept

The inspiration to create the 'Box Of Chops' rose from the laborious way of creating vocal chops within a DAW. Often it is not intuitive, and the time taken to create the idea heard in your head can put a stop to a chain of creative thoughts. There is a need to create a product which allows an intuitive, fast and creative way to create vocal chops.

Personally, I am a fan of vocal chops being used in songs because of the interesting sonic qualities they can bring to a track. An example of a song which uses vocal chops to a great extent is 'Turn Down for What'- DJ Snake and Lil Jon (Snake and Jon, 2013). As a producer, I believe that the near infinite choices a DAW offers to manipulate audio can be overwhelming and counteractive to creativity. Experts say that 'imposing restrictions is more likely to encourage creativity.' (Perry, 2022), as is evident from Miles Davis 'Kind of Blue' album (Davis, 1959), where he restrained himself to not play one chord across the entire album. Creating a product which hones in on the necessary features to create vocal chops and places restrictions could be helpful in inspiring ideas.

For a user with little knowledge of music theory and piano playing skills, the idea of creating a vocal chop melody on a midi keyboard could be daunting and limiting to creativity. Even for experienced producers, when faced with a keyboard, often muscle memory results in playing

the same old melodies. Designing a way to make creating new melodies greatly more inspiring would allow users to break out of a creative rut.

Analysing a current commercial vocal sampling product, The Waves CR8 creative sampler (Waves, 2022) appears to be a useful tool to manipulate vocal chops, boasting a vast number of features to create unique sounds. However, it is only available as a plugin, and whilst vocal chops are able to be played via a midi keyboard it ties the user into using a DAW. Creating a product which can remove the user from a DAW forces a different work flow which is inspiring.

Another commercial vocal sampling product- The Teenage Engineering PO-35 SPEAK (Teenage Engineering, 2022), is interesting due to its handheld portable design. In terms of creativity, being hands on with physical switches and knobs is superior to using a mouse and keyboard with a DAW. The PO-35 offers many features in addition to vocal chopping which although impressive, as discussed previously, could be limiting to creativity. Producing a physical standalone vocal chopping product with intended constraints is intriguing and offers value.

Both commercial products focus so much on offering a large number of features that they neglect the specifically useful features that would make vocal chop production easier and more intuitive.

Project outline: Ideation

After having researched current commercial vocal samplers, there appears to be a gap in the market to focus on creating a simple and inspiring vocal chop device.

The 'musts' for my product are:

My product must be a battery powered handheld design to be portable. It must stay away from relying on a midi keyboard and use a unique way to play melodies. It must simplify the controls to a play/stop button and a sample selection selector. To make it simple to play along with any track, a feature to choose the key and tonality would be useful. Doing this would separate the theoretical and creative side of melody creation which would make the user use their ears more than their head, opening up new possibilities.

It must provide an inspiring, unique, quick, easy and accessible way to create new melodies. In terms of design, my product should be visually eye catching and minimal to make users want to pick it up and use it without second guessing how anything works.

My thoughts were that although this product may negate the need to know any music theory, it could serve as an educational tool too. If a joystick could be used, the number positions of the joystick could translate to the number degrees of a musical scale. Making a connection

between the two would mean the user could learn what note number patterns create a particular emotion within a key.

Keeping the features limited is important, but the integration of a select number of effects such as formant and distortion would increase the sound possibilities of the product. Unfortunately, this was found to be beyond the scope of this project due to the complexity of introducing these features, but it could be bared in mind and considered when developing the product further.

Prototyping: Research

After the scope of the project was defined and there was a clear direction which the product was heading, research was required to discover which components would be suitable to accomplish the aims set out.

An Arduino microcontroller was found to be useful in this situation due to the ease of connecting inputs and outputs and writing code to carry out functions. To make the product portable, audio samples must be stored in the product. The Arduino MKR Zero is designed for music playback and contains a micro SD card slot. It also utilises the I2S protocol which follows the 'Philips standard' for connecting digital audio devices together.

The stereo I2S DAC breakout board integrates seamlessly with the Arduino board via the I2S protocol to convert the digital signal into an analog stereo line level output.

A 16 position BCD (Binary coded decimal) rotary encoder was chosen to act as a key selection feature. Due to the Arduino board having a limited number of inputs, the BCD switch only has 4 outputs. It works by outputting highs and lows on specific pins depending on the switches position. In the code, if statements were setup for each of the 16 combinations to decide what to do. As there are only 12 musical keys, 4 positions were left over. However, these have been used as a bonus feature for drum sample playback.

A capacitive touch sensor was chosen for the play button. It outputs a high when it senses the capacitance from a finger. This was chosen instead of a mechanical button because it allows faster re-triggering of samples. It is also an interesting USP of the product.

An 8-position joystick was chosen to select the sample to be played. The joystick works by outputting high on the pin which corresponds with the joystick position. A joystick was chosen because it's unique, fun and engaging. It also makes the device feel like a handheld games console.

A slide switch chooses between major or minor. It is simple and serves its function.

A stereo ¼" jack socket was chosen to serve as the audio output. It is an industry standard connection and can easily be integrated with other studio equipment.

A rocker switch was selected for powering on or off the device. It is a sturdy construction and is fool proof.

To make the product portable, a 150mAh LiPo battery will provide more than enough current for the circuit and will have a long battery life.

An enclosure was chosen which would be ergonomic in the sense that it fits comfortably in the hands, and is also physically large enough to house the components.

Once a parts list was created, an order was placed.

Prototyping: Construction

As soon as the parts arrived, I drew a sketch of where the components would be positioned (See Appendix B). I ensured the touch sensor and joystick were positioned ergonomically and naturally where the thumbs rest. Once the sketch was created, this was translated to the physical box where the centre positions were marked.

Fortunately, my Dad runs an engineering business which meant I was able to use his machinery including a milling machine, a bandsaw and a lathe. The holes were drilled out and a sub chassis was constructed to fix the internal components to (See Appendix C).

Although the enclosure chosen was the correct size, using a 3D printer would have allowed for a custom case to be designed which could have been slightly shallower and include contours to make it more comfortable to hold in the hands. Injection molding is another useful manufacturing process, which is 'commonly used to quickly and reliably produce high-volume runs' (McGuigan, 2022). In my scenario, since I only have the need to produce one casing at the moment, 3D printing would be more cost effective. Also, if adjustments were needed, with a 3D printer it would be much quicker to produce than with injection moulding.

Once all of the components were housed in the enclosure, the wiring and soldering process began (See Appendix D). Ensuring that not too much heat was applied to components from the soldering iron was challenging and doing so could cause irreversible damage. Soldering small wires to header strips also proved challenging and required patience and a steady hand. It was important to ensure there were no dry joints or any blobs of solder touching other pins. Upon checking, I discovered two places where the solder had flowed across a track, which could have been extremely problematic if it hadn't been spotted.

Prototyping: Coding and initial testing

With everything connected and checks carried out, the challenge of writing the code began (See Appendix E for complete code).

Arduino microcontrollers use the open source software 'Arduino IDE' and use the coding language C++. I have some previous experience of coding and know the basic principles. For example, if, else and while statements, but the code required for this project is much more complex. Not only is it necessary to write the code to carry out operations, with the addition of the SD card and I2S protocol specific code is needed to make these features work and talk to each other. Specific libraries were also necessary to download in Arduino IDE to run alongside the code. The code to initialize the SD card and I2S were sourced from various Arduino (Arduino, 2022) and Git Hub (The Inter-IC Sound (I2S) Protocol, 2022) pages with some modification to make them fit my specific needs.

Due to the complexity of the coding, there were many issues encountered. Fortunately, with my Dad also having some knowledge of coding (although not in C++), he was able to give me pointers when I was faced with an issue.

After realizing that the switches were not responding to High and Low commands, it was eventually found that the pins of the Arduino were held high by pull up resistors. This was discovered by finding schematics of the Arduino board and seeing which pins had internal pull up resistors connected. To combat this, I chose to pull the pins down to ground by referencing the switches to ground rather than to 3.3v. This means that the Highs and Lows in the code are reversed.

Initially I faced issues with making the DAC board communicate with the Arduino. The code to setup the I2S communication did not state which pins to connect. Researching deeper into the associate installed library found which were the correct pins and the issue was resolved.

To make the touch sensor play and stop the sample, initially I tried to create a while loop. This did not work, and I assumed it to be a limitation of the functions in the particular library. To resolve this, I decided to try further if statements and remove the while loop, and this resolved the problem.

When testing the outputs of the joystick, it appeared that one pin was not functioning as there was no High or Low output coming from it. Upon inspection I found that the header pin was not inserted fully into the Arduino, so the connection was incomplete.

A big hurdle I came across was that the drum samples were distorted when played back. There was no logical reason why this was happening as the code looked correct and the vocal samples were playing back perfectly. Via trial and error, it appears that the samples were too short in length (less than 1 second) and this may have been causing the clock signal to the stereo

decoder breakout board to go out of sync. To combat this, I re-bounced the samples with silence at the end to make each sample 2 seconds long. I was pleased to discover this had fixed the problem.

After resolving these issues, the instrument operated as expected.

Visual branding design

After completing the instrument build and knowing everything was in the right place and wasn't going to be moved, I began to think about the visual elements and how they would complement the hardware features.

I decided to make the design fun and eye catching to tie in with the brand idea of being modern and unique. I improved on my initial design sketch by making the artwork more fun and captivating (See Appendix F for the final digital artwork design).

Comparison to commercial products

The NI Maschine+ hardware unit (MASCHINE+, 2022) is very impressive. It has a vast number of features available to create some truly great sounds. However, with regards to the playback of vocal chops, it is fairly limited with what it can do. To play vocal chops, you would have to assign samples to each pad, taking time to work out what pitches the samples need to be for each pad to fit the key of your song. With my product, you simply have to rotate the switch and the work is done automatically. A strength of the Maschine+ is that you can change the velocity of the samples with their pads, whereas with my product the touch sensor is one velocity only. This is something I could develop in the next iteration of the product.

Comparing my product with the Waves CR8 sampler, clearly the CR8 has more control over ADSR manipulation and contains effects such as adjusting the speed and filter cutoff of the samples. However, it is software based only, and can only be controlled via a midi keyboard. This limits the audience to users who only have a DAW and operating system capable of running this plugin. With my product being standalone, anybody can use it.

Whilst the PO-35 contains more features than my product such as an inbuilt microphone, it is not as easy to quickly change between stored samples, therefore making vocal chop melodies more difficult. Also, at first glance it may look complicated to use for some users, which might put them off investing money into it. On the other hand, the simplicity of my design makes it appealing to all age groups.

Beta testing and user feedback

It is vital to receive user feedback on a new product. The feedback given can be used when creating the next revision of the product in the hope that it will be an improvement over the previous design.

My product was tested by 3 individuals- User 1 being a producer, User 2 being a drummer and User 3 being a non-musician. All 3 users commented on how much they liked the joystick, stating that it felt like a gaming controller. User 2 having a passion for gaming also said that the joystick could be improved by having a feeling similar to the one found on a Game Cube controller.

User 1 expressed how it could be improved through the inclusion of a built-in speaker. They said that it was disappointing not to be able to use it completely on its own, as without a speaker you need to plug it into an interface to hear it.

With user 3 being a non-musician, I was pleased to hear that he enjoyed the user interface, and had fun using it.

After personally beta testing the product, I found that the touch sensor can sometimes be temperamental. Looking into this issue further, it seems that after the device has been turned on for a while, the sensor becomes less sensitive. Potentially it could be an issue with the code and is something I will have to look into further as part of product development.

Reflection and further development

Reflecting on the success, weaknesses and further improvements I could make to my product, I was happy to find that it met most of the criteria set out.

The hope that it could be used as an educational tool proved true. For example, after creating a melody, if you recall the number positions of the joystick used you can translate these into the note degrees of the scale. For example, playing positions 1 (up on the joystick), 3 (left on the joystick) and 5 (down on the joystick) will create a root note major or minor triad in the chosen key. This can be educational as the user begins to relate the joystick position numbers to the note degree numbers of the scale. For beginners in music this could be helpful to learn how to create melodies and relate them to note degrees in a key. It could also help experienced producers to improve their music theory knowledge by familiarizing the way note patterns sound and the emotional qualities they produce within a key.

With my product being battery powered and portable, it can be used away from a desk and passed around the room for everyone to use. This is a big USP of the product.

The ability to cut samples short, e.g. a crash sample creates an interesting approach to melody creation and also drum programming.

The ergonomics of the product are successful in terms of being comfortable to hold in the hands.

The use of the capacitive touch sensor and joystick make it feel like you are playing a game, which makes it fun and easy to create vocal chops. These unique operating methods will allow the user to escape being stuck in a rut of familiarity with a keyboard and be inspired to create melodies intuitively and think outside of the box.

Currently, in order to charge the battery, the power switch has to be in the on position.

To improve my product, I could look into adding effects such as formant shifting and distortion to add sonic diversity. To do so would require extensive research into further coding and inclusion of other breakout boards.

Another improvement would be to make the touch sensor velocity sensitive, so that the samples can be played at different velocities. This would require additional coding and more samples.

I would improve the operation of the joystick, since it doesn't have much accuracy for repeatability. Finding another joystick which has movement limited to the 8 axes only would make it easier to consistently play the same melodies.

Also, an improvement would be to enter another mode when the joystick is clicked in, which would pause the samples when the touch sensor is released rather than stop them. Therefore, when the touch sensor is pressed again, the samples will continue to play from where they left off. This would add more variety to the sounds achievable from the device. Also, allowing sample change to happen whilst the touch sensor is held would allow a further variety of sounds to be created.

Allowing the user to change the key seamlessly during performance would be an interesting improvement. At the moment it is impossible to do with two hands since the touch sensor and joystick both need to be in operation in order for a sound to be played. The code would have to be modified so that the sample can change whilst the touch sensor is pressed.

Occasionally there is some mild interference noise, which is probably due to a lack of shielding. In development, all audio signal cables would be properly shielded.

An alternative version of this product could be to have 16 different banks of instrument samples, so that it is not limited to vocal sampling only. This would be simple to do and would only require uploading the samples to the SD card and a minor modification to the code.

As an improvement, I would replace the toggle on the joystick with a thumb pad for extra comfort and ease of use.

A further development would be to have the capability to plug in a microphone and record samples directly to the SD card which could then be played in real time. This would require extensive research into how to make it possible with coding.

A simple yet effective improvement would be to include a volume knob on the output. This would give further control for the user.

In the next development stage, it would be useful to test the battery life so that it could be stated in the specifications on a website once it is for sale.

Conclusion

To conclude, I am very pleased with the outcome of my product. It operates as I intended it to. Although I faced some difficult moments in the creation process, I have gained a huge amount of knowledge regarding coding, and I discovered that I enjoyed the problem-solving element that coding brings. I am looking forward to continuing the development of this project and seeing where it could lead.

Bibliography

Arduino Official Store. 2022. *Arduino MKR ZERO (I2S bus & SD for sound, music & digital audio data)*. [online] Available at: <<https://store.arduino.cc/products/arduino-mkr-zero-i2s-bus-sd-for-sound-music-digital-audio-data?selectedStore=eu>> [Accessed 25 March 2022].

Arduino.cc. 2022. *Arduino - I2S*. [online] Available at: <<https://www.arduino.cc/en/Reference/I2S>> [Accessed 25 March 2022].

Davis, M., 1959. *Kind Of Blue*. [Digital] Columbia. Available at: <<https://open.spotify.com/album/1weenld61qoidwYuZ1GESA?si=6FyRndyVRzuPPSqvZADmUQ>> [Accessed 25 March 2022].

Docs.arduino.cc. 2022. [online] Available at: <<https://docs.arduino.cc/learn/programming/sd-guide>> [Accessed 20 March 2022].

McGuigan, D., 2022. *Plastic Injection Molding vs 3D Printing – Which is Better?*. [online] Kaysun.com. Available at: <<https://www.kaysun.com/blog/plastic-injection-molding-vs-3d-printing>> [Accessed 25 March 2022].

Native-instruments.com. 2022. *MASCHINE+*. [online] Available at: <<https://www.native-instruments.com/en/products/maschine/production-systems/maschine-plus/>> [Accessed 25 March 2022].

Perry, P., 2022. *Why Imposing Restrictions Can Actually Boost Creativity*. [online] Big Think. Available at: <<https://bigthink.com/mind-brain/why-imposing-restrictions-can-actually-boost-creativity/>> [Accessed 25 March 2022].

Shop.pimoroni.com. 2022. *Adafruit I2S Stereo Decoder - UDA1334A Breakout - Pimoroni*. [online] Available at: <<https://shop.pimoroni.com/products/adafruit-i2s-stereo-decoder-uda1334a-breakout?variant=1344798982154>> [Accessed 25 March 2022].

Shop.pimoroni.com. 2022. *Adafruit Standalone Momentary Capacitive Touch Sensor Breakout - Pimoroni*. [online] Available at: <<https://shop.pimoroni.com/products/adafruit-standalone-momentary-capacitive-touch-sensor-breakout?variant=334997573>> [Accessed 25 March 2022].

Shop.pimoroni.com. 2022. *LiPo Battery Pack - Pimoroni*. [online] Available at: <<https://shop.pimoroni.com/products/lipo-battery-pack?variant=20429081991>> [Accessed 25 March 2022].

Snake, D. and Jon, L., 2013. *Turn Down for What*. [Digital] Columbia. Available at: <<https://open.spotify.com/track/67awxiNHNyjMXhVgsHulrs?si=6d419fbdc84043df>> [Accessed 25 March 2022].

Sound Examples. 2022. *The Inter-IC Sound (I2S) Protocol*. [online] Available at: <<https://tigoe.github.io/SoundExamples/i2s.html>> [Accessed 20 March 2022].

teenage engineering. 2022. *PO-35 speak*. [online] Available at: <<https://teenage.engineering/store/po-35/>> [Accessed 25 March 2022].

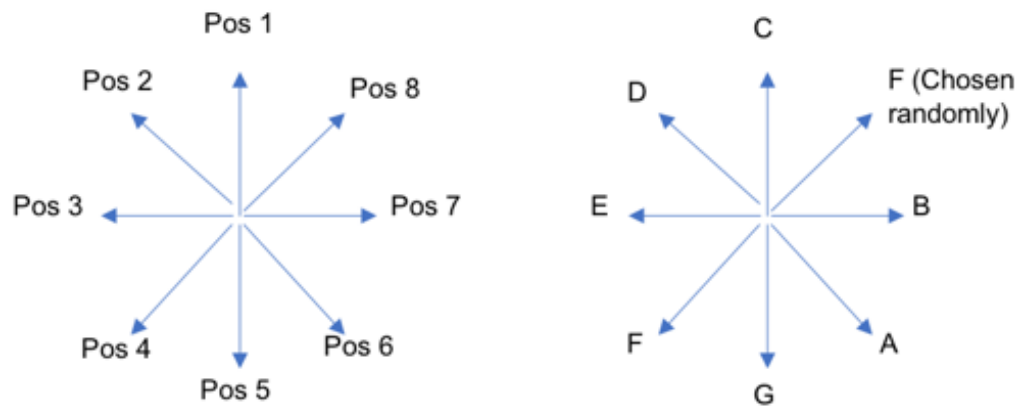
Uk.rs-online.com. 2022. *BCKS1002RS | Lorlin, 16 Position, BCD Rotary Switch, 150 mA @ 250 V ac, Through Hole / RS Components*. [online] Available at: <<https://uk.rs-online.com/web/p/rotary-switches/7392924>> [Accessed 25 March 2022].

Uk.rs-online.com. 2022. *RKJXL100401V / Alps Alpine Center-Push, 8-Axis Joystick Switch Lever, Momentary / RS Components*. [online] Available at: <<https://uk.rs-online.com/web/p/joystick-switches/1239634>> [Accessed 25 March 2022].

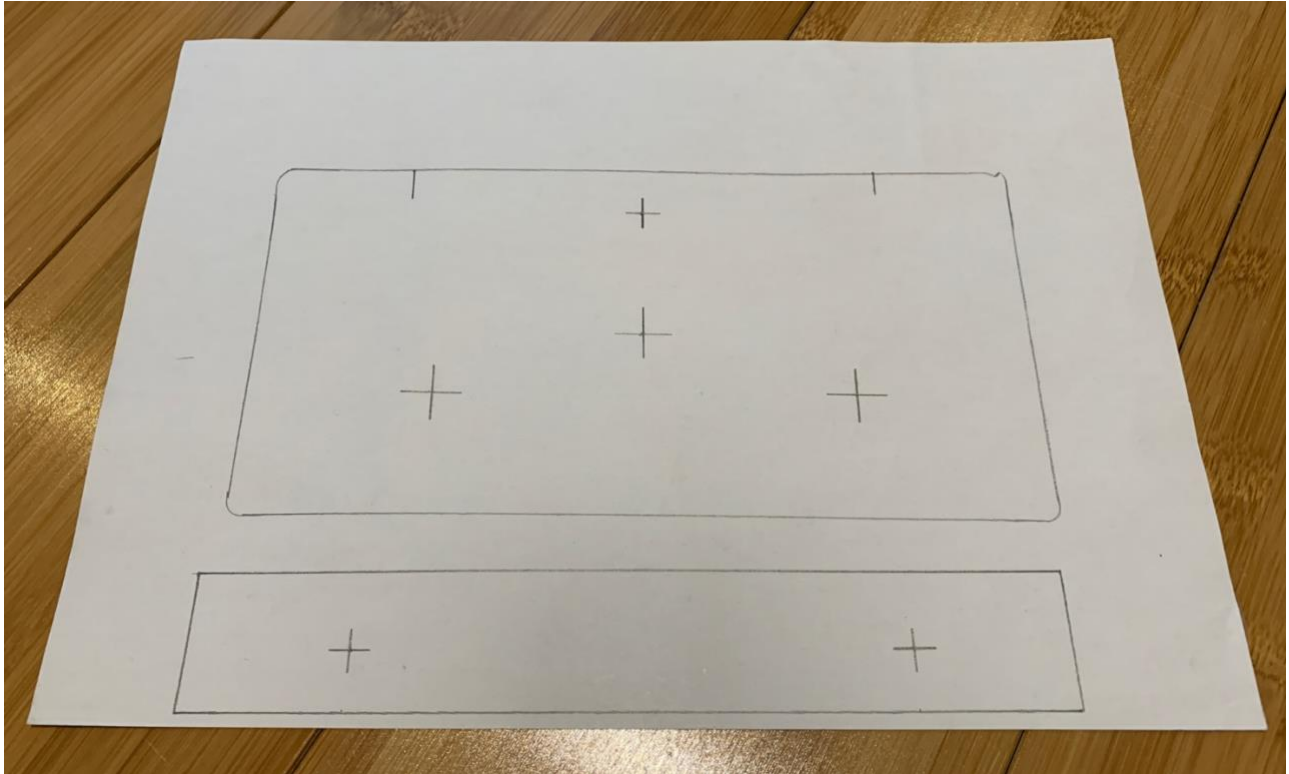
Waves.2022. [online] Available at: <https://www.waves.com/plugins/cr8-creative-sampler?w_campaign=1484676631&gclid=Cj0KCQjw_4-SBhCgARIsAAlegRWse3g5URPYL7aT9goN3vIw5kscnSp4MeY8DdlEht9J6F2AJ6Kt5VYaAkIBeALw_wcB#introducing-cr8-creative-sampler> [Accessed 25 March 2022].

Appendix A

Example showing how the notes of the C major scale are mapped to each joystick position.

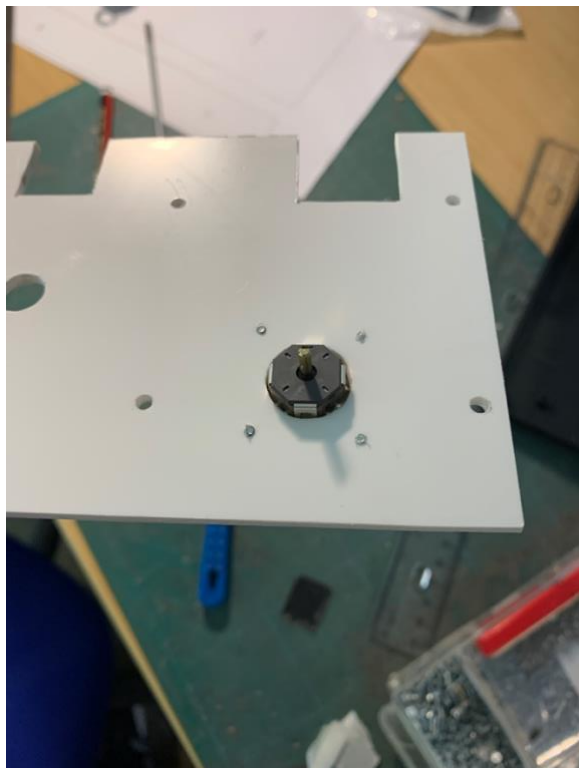


Appendix B



Appendix C





Appendix D



Appendix E

```

v20_vocal_chopper
1 |
2 #include <SD.h>
3 #include <ArduinoSound.h>
4
5 const char SFXA[] = "SFXA.WAV";
6 const char SFXAS[] = "SFXAS.WAV";
7 const char SFXB[] = "SFXB.WAV";
8 const char SFXC[] = "SFXC.WAV";
9 const char SFXCS[] = "SFXCS.WAV";
10 const char SFXD[] = "SFXD.WAV";
11 const char SFXDS[] = "SFXDS.WAV";
12 const char SFXE[] = "SFXE.WAV";
13 const char SFXF[] = "SFXF.WAV";
14 const char SFXFS[] = "SFXFS.WAV";
15 const char SFXG[] = "SFXG.WAV";
16 const char SFXGS[] = "SFXGS.WAV";
17 const char SAYNO[] = "SAYNO.WAV";
18 const char KICK[] = "KICK.WAV";
19 const char SNARE[] = "SNARE.WAV";
20 const char HATC[] = "HATC.WAV";
21 const char HATO[] = "HATO.WAV";
22 const char CRASH[] = "CRASH.WAV";
23 const char COW[] = "COW.WAV";
24 const char TOM[] = "TOM.WAV";
25 const char CLAP[] = "CLAP.WAV";
26
27 const int touchState = 6;
28 const int joystickA = 14;
29 const int joystickB = 13;
30 const int joystickC = 12;
31 const int joystickD = 11;
32 const int joystickE = 10;
33 const int joystickF = 9;
34 const int joystickG = 8;
35 const int joystickH = 7;
36 const int rotaryA = 0;
37 const int rotaryC = 1;
38 const int rotaryD = 4;
39 const int rotaryB = 5;
40 const int majMinSwitch = 19;
41
42 SDWaveFile waveFileA;
43 SDWaveFile waveFileAS;
44 SDWaveFile waveFileB;
45 SDWaveFile waveFileC;
46 SDWaveFile waveFileCS;
47 SDWaveFile waveFileD;
48 SDWaveFile waveFileDS;
49 SDWaveFile waveFileE;
50 SDWaveFile waveFileF;
51 SDWaveFile waveFileFS;
52 SDWaveFile waveFileA;
53 SDWaveFile waveFileAS;
54 SDWaveFile waveFileB;
55 SDWaveFile waveFileC;
56 SDWaveFile waveFileCS;
57 SDWaveFile waveFileD;
58 SDWaveFile waveFileDS;
59 SDWaveFile waveFileE;
60 SDWaveFile waveFileF;
61 SDWaveFile waveFileFS;
62 SDWaveFile waveFileCL;
63
64 void setup() {
65   Serial.begin(9600);
66
67   pinMode(touchState, INPUT);
68   pinMode(joystickA, INPUT_PULLUP);
69   pinMode(joystickB, INPUT_PULLUP);
70   pinMode(joystickC, INPUT_PULLUP);
71   pinMode(joystickD, INPUT_PULLUP);
72   pinMode(joystickE, INPUT_PULLUP);
73   pinMode(joystickF, INPUT_PULLUP);
74   pinMode(joystickG, INPUT_PULLUP);
75   pinMode(joystickH, INPUT_PULLUP);
76   pinMode(rotaryA, INPUT_PULLUP);
77   pinMode(rotaryC, INPUT_PULLUP);
78   pinMode(rotaryD, INPUT_PULLUP);
79   pinMode(rotaryB, INPUT_PULLUP);
80   pinMode(rotaryD, INPUT_PULLUP);
81   pinMode(majMinSwitch, INPUT_PULLUP);
82
83   Serial.print("Initializing SD card...");
84   if (!SD.begin()) {
85     Serial.println("SD card initialization failed!");
86     return;
87   }
88   Serial.println("SD card is valid.");
89 }
90
91 void loop() {
92   if (touch == HIGH) {
93     if (AudioOutI2S.isPlaying()) {
94       //Serial.println("playing");
95     } else {
96       if ((rotA == LOW) && (rotB == HIGH) && (rotC == HIGH) && (rotD == HIGH)) //A
97       {
98         if (joyA == LOW)
99         {
100           if (majMin == LOW)
101           {
102             AudioOutI2S.play(waveFileA);
103           }
104           else
105           {
106             AudioOutI2S.play(waveFileAS);
107           }
108         }
109         if (joyB == LOW)
110         {
111           if (majMin == LOW)
112           {
113             AudioOutI2S.play(waveFileB);
114           }
115           else
116           {
117             AudioOutI2S.play(waveFileBS);
118           }
119         }
120         if (joyC == LOW)
121         {
122           if (majMin == LOW)
123           {
124             AudioOutI2S.play(waveFileC);
125           }
126           else
127           {
128             AudioOutI2S.play(waveFileCS);
129           }
130         }
131         if (joyD == LOW)
132         {
133           if (majMin == LOW)
134           {
135             AudioOutI2S.play(waveFileD);
136           }
137           else
138           {
139             AudioOutI2S.play(waveFileDS);
140           }
141         }
142       }
143     }
144   }
145 }

```

```
v20_vocal_chopper
226
227     }
228     if (joyH == LOW)
229     {
230         if (majMin == LOW)
231         {
232             AudioOutI2S.play(waveFileE);
233         }
234         else
235         {
236             AudioOutI2S.play(waveFileE);
237         }
238     }
239 }
240
241 }
```

The above code repeats 16 times for each of the 16 rotary switch positions. To save space in this document, the below image skips to the point in the code after the 16th repeat.

```
2
3     if ((rotA == HIGH) && (rotB == LOW) && (rotC == LOW) && (rotD == LOW)) // pos 14
4     {
5         AudioOutI2S.play(waveFileSN);
6     }
7     if ((rotA == LOW) && (rotB == LOW) && (rotC == LOW) && (rotD == LOW)) // pos 15
8     {
9         AudioOutI2S.play(waveFileSN);
10    }
11
12
13    }
14
15    if (touch == LOW)
16    {
17        AudioOutI2S.stop();
18        //Serial.println("stopped");
19    }
20 }
```

The above image shows the last lines of the code.

Appendix F

