‘You spin me right round’ drum machine (Fergus Jack Doherty-Eagles 260852322)

Introduction

After the Arduino section of the course, I was inspired to try to use whatever I could find in the Super Starter Kit to create some kind of musical device. I liked the idea of making something tangible that could be manipulated by the user in real time, and that had some unique functionality as a result of its interactive design - a functionality that would be hard to simulate digitally.

One particular class I was looking at the motor, and thought that it could be interesting to use it in a rotating drum machine of sorts. The periodicity of rotation is inherently musical (after all, a sinusoid is just a trace of the height of a point as it travels around a circle), and I thought that one rotation could quite intuitively map to one ‘measure’ of music.

With this, I sketched the first diagram of a potential implementation.

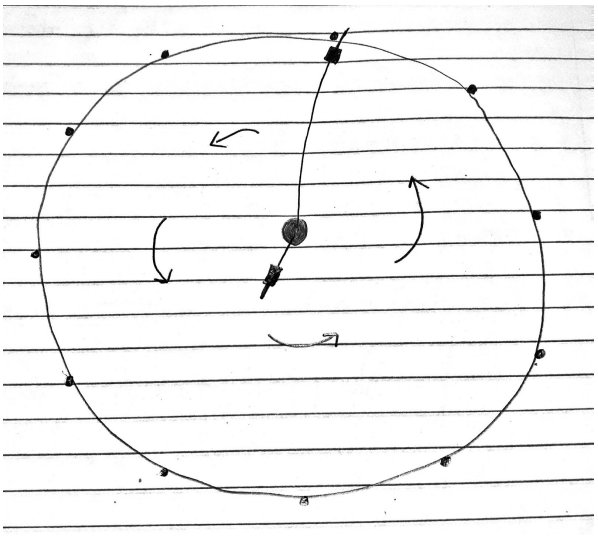


Figure 1: Initial brainstorming diagram of the machine

My idea was to connect a wire from the Arduino Kit to a rotating centre, and have one end ‘flick’ pins sticking vertically upwards on the outside of a circular base. The momentary connection would bring about some measurable change in a circuit, which I could then use to trigger drum sounds. Each pin could be hardwired to correspond to a different drum sound, perhaps by assigning distinct resistor arrangements to distinct pins*. The advantage of this would be that the user could make micro timing adjustments, to give good, intuitive control over the ‘feel’ of the drum loop. It would also look really cool.

* As it happened, this was hard to implement, so I went with a different strategy. But not much else changed in the final prototype, as the rest of the report will outline.

Construction

The first hurdle in building this machine was figuring out how to maintain an electrical connection with spinning components. The smaller half of the rotating pin (see diagram above) had to somehow be permanently in contact with the rest of the circuit. Initially I was planning to lay down a sheet of aluminum foil over whatever circular base I found. However, I stumbled

upon a cocoa powder tin in the recycling which was PERFECT, since it was circular, about the right size, and, crucially, had a metal bottom! In hindsight, this was extremely helpful.

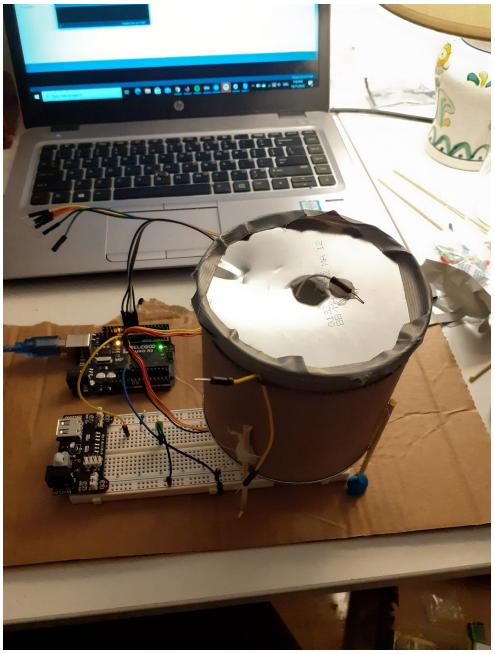


Figure 2: The cocoa powder base in its position. I used hot glue and skewers to raise it, so that wires could be passed under and I could still see the motor

Figure 3: A close up the hole punched through the base, and the hot glue used to connect a wire from the metal plate to the rest of the circuit

The motor I used was a stepper motor that came in the kit - usually stepper motors are meant for precise movements, not so much continuous rotation. However, I did try the other motor in the Elegoo kit, but its slowest speed was much too fast, so I opted for the stepper. The problem with the continuously running stepper was that it drained 9V batteries in less than a minute, but I was able to purchase a small external power supply from Amazon which solved the problem.

Next I built a rotating centre which was going to be connected to the motor. A small amount of bluetack and a small surrounding ‘column’ of tape was a suitable mounting point. I also glued some shredded up aluminum foil to one end of the pin to act as a ‘brush’ contact.

In order to keep all of the various components from being chaotically disorganized (for example, the ribbon cable used for the stepper motor driver was a real nuisance), I used putty to stick everything to a cardboard base.

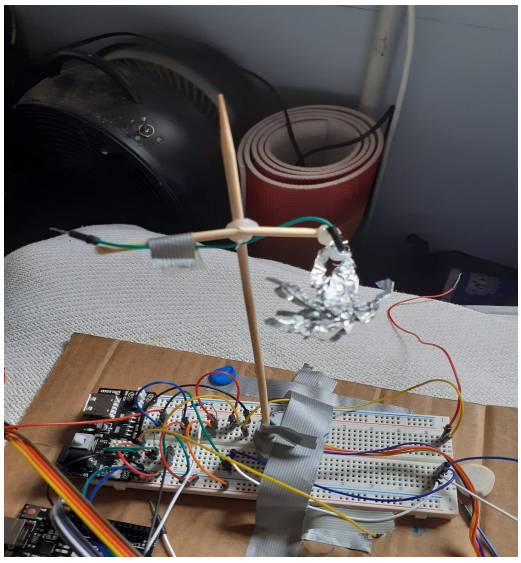
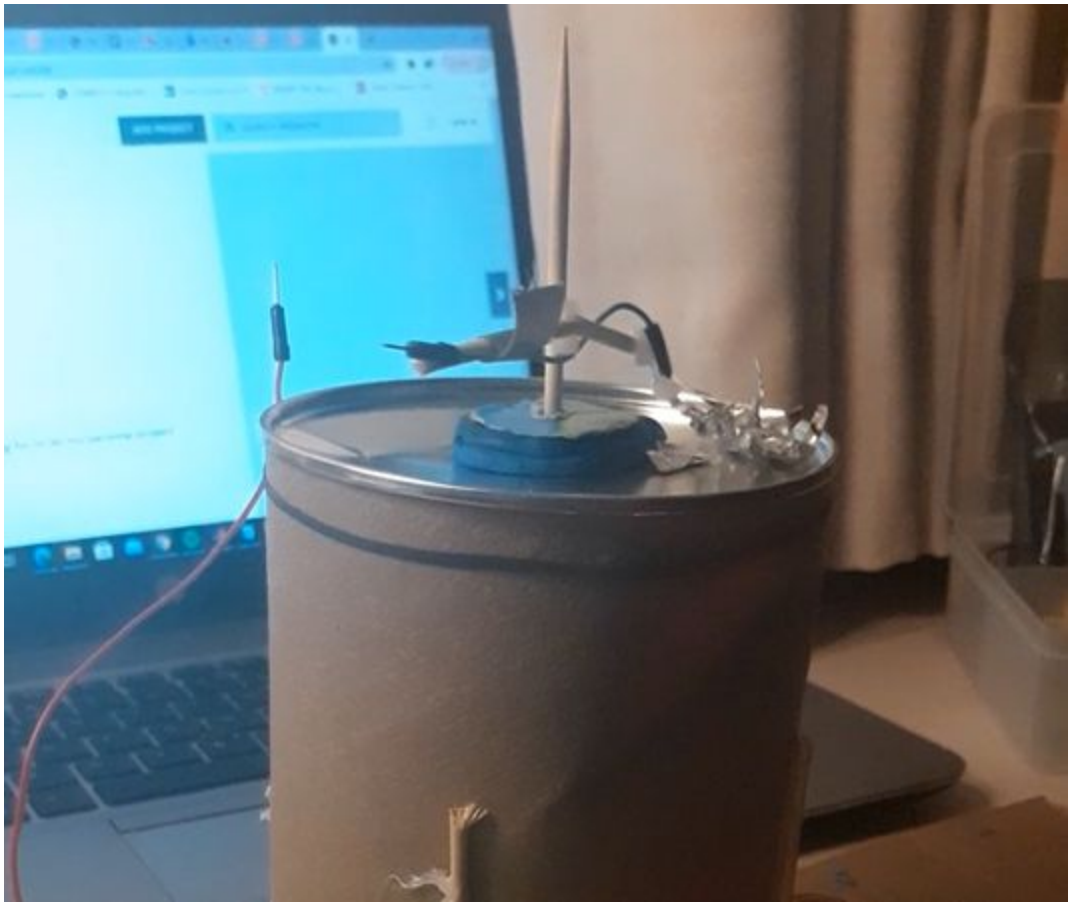


Figure 4: Rotating centre. The tape acted as a ‘container’ for the bluetack which was the main thing that stuck the skewer to the motor

Once I had all of these components, I needed to add the wires on the exterior. I held them up with an elastic band, which meant I would be able to slide them minute distances around the exterior of the base. This would enable me to get the ‘micro’ adjustments required to have a good control over the groove. Below is an example with just one wire for clarity.



As seen in the above diagram, I also used a small amount of blu-tack and a small cardboard cut out to narrow the hole and ensure that the axis of rotation of the central skewer was almost perfectly vertical.

Circuitry

I had planned to hardwire the circuit so that different pins would trigger different drum sounds automatically (that is, the measured change in the voltage would be distinct for different sounds). However, I never figured out a way to do this, since there was only one cable coming from the device (the white one in the images above), so I couldn't differentiate between which pin was being hit. As a result, I changed the circuit design to just register when any pin had been 'flicked', and planned to do the manipulation of the different sounds in Max.

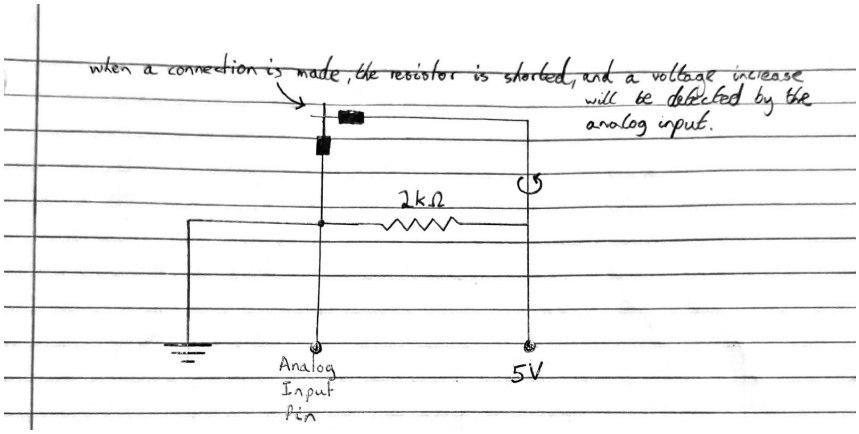


Figure 5: Schematic diagram of the circuit

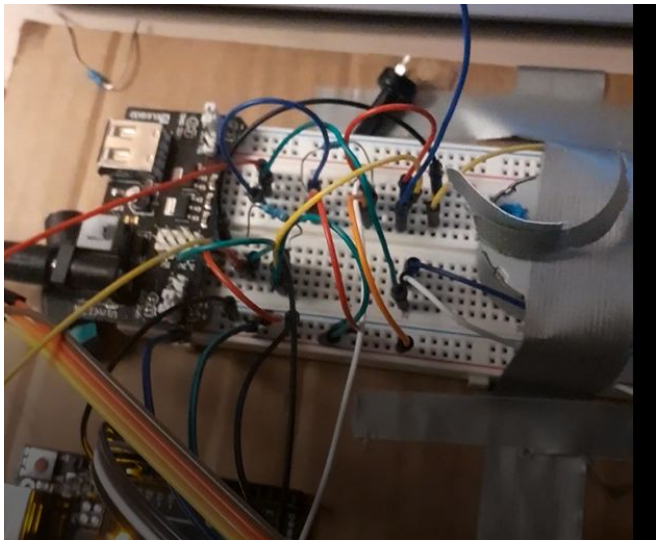


Figure 7: The wired circuit. It looks more complicated than it really is because I did not remove my initial attempt with hardwiring the two distinct pins

Arduino Code

```
#include <Stepper.h>
int boomPin = A2;
int kaPin = A1;
int boomvalue;
int kavalue;
int delayT = 20;
int motSpeed = 12;
int stepsPerRevolution = 2048;
Stepper myStepper(stepsPerRevolution, 8, 10, 9, 11);

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  myStepper.setSpeed(motSpeed);
}

void loop() {
  // put your main code here, to run repeatedly:
  myStepper.step( 17 );
  boomvalue = analogRead(boomPin);
  kavalue = analogRead(kaPin);
  Serial.println( kavalue );

  //delay ( delayT );
}
```

In the first part of the arduino code, I declare some useful variables.

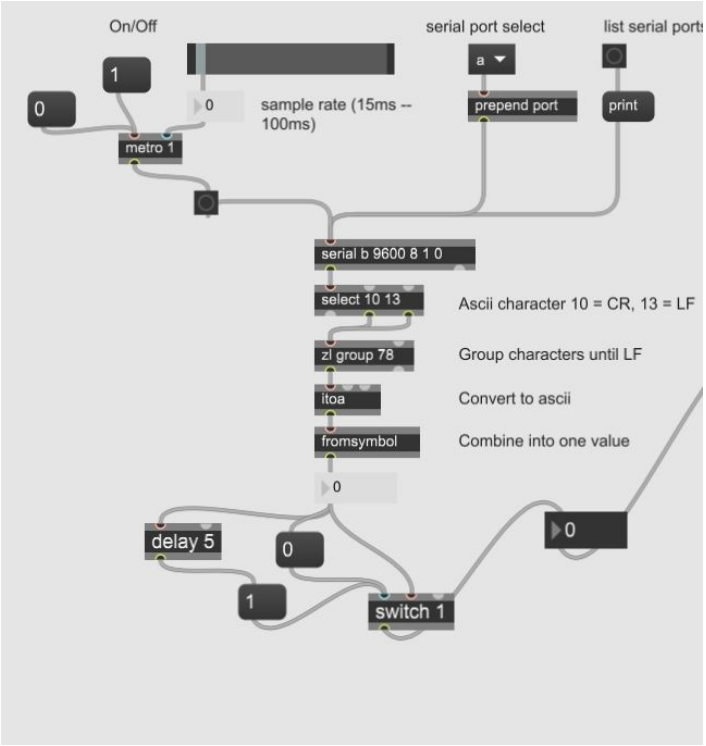
Begin the serial!

Everytime the loop executes, it does 17 steps on the motor (by trial and error, I found this was the best way to stop any ‘shuddering’, while still being fast enough.

It is worthy to note that there are apparently two values being read in this code . This is again related to my initial attempt at the implementation with the distinctly hardwired sounds. In fact, only one value was required, which is why I only print the ‘kavalue’ to the serial (the ‘kavalue’ also changes when a ‘boom’ pin is flicked). I just didn’t fully change the code.

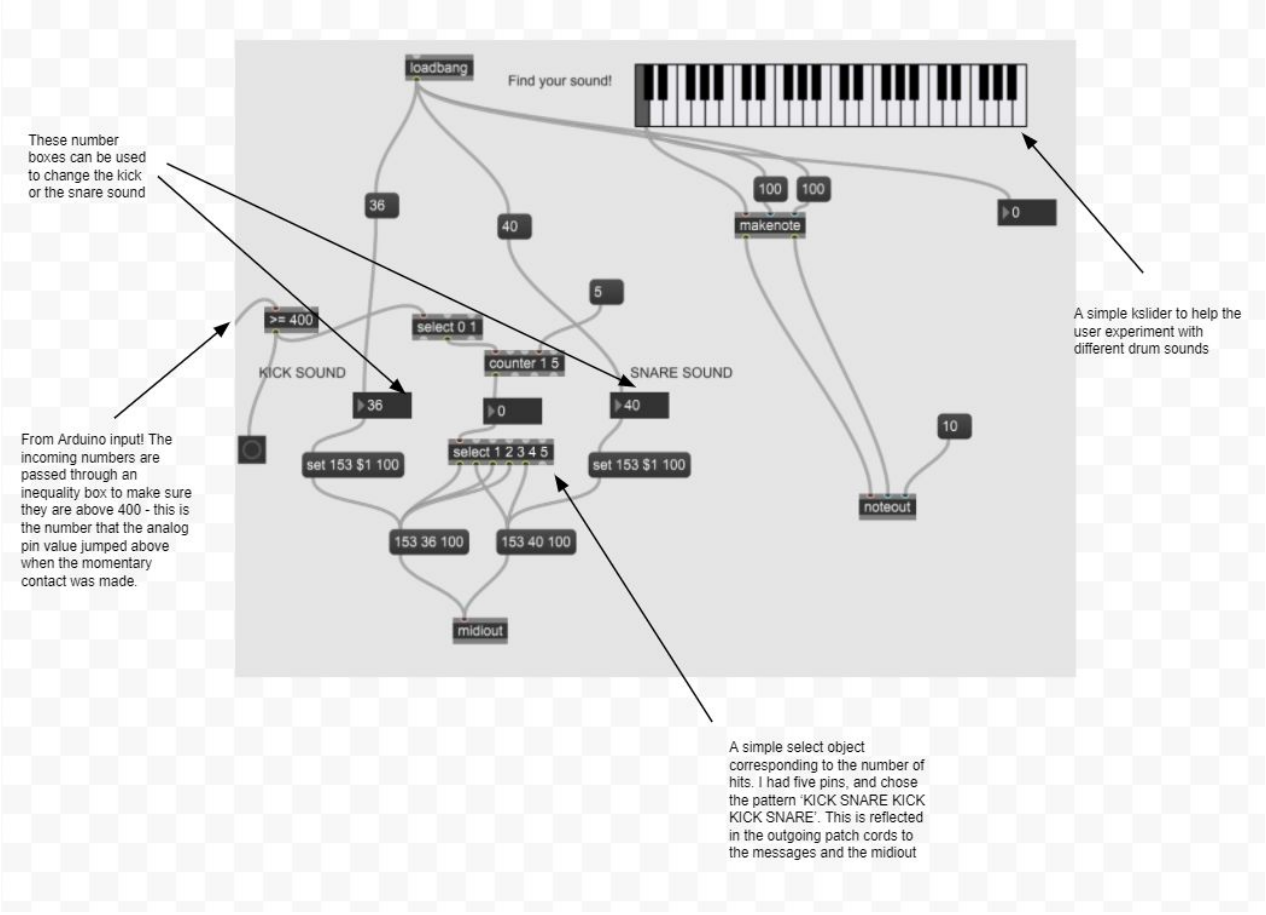
Max Patch

The Max Patch had to be able to trigger a sound whenever a pin was flicked. I used the patch we looked at in class to read values from the Arduino, and added some functionality to set up a specific drum pattern. The number of hits in the ‘select’ object corresponds to the number of pins, and each hit triggers either a kick or a snare sound.



The top part of this code is exactly the same as the code we saw in class.

I encountered a problem that sometimes a pin would maintain contact long enough with the rotating centre wire that more than one value would be read from the serial, and it would register as two or more pin flicks. To remedy this, I added a small timed switch, so that no more values would be read for the next 5ms after one value was already read.



Links to successful operation of the device

I had some success using the device! Here are some links to it in use.
Link 1: First successful operation
https://www.youtube.com/watch?v=I4hMVpWm6LI
Link 2: Demonstration of changing the programmed sounds using the max patch
https://www.youtube.com/watch?v=SUt6XauGZ2Q

Problems encountered

The main problem I faced was actually manipulating the pins when they were attached to the side of the cocoa powder tin. The device was held together largely by duct tape and hot glue, so it was not particularly robust or stable. Whenever I moved one pin, I would have a hard time getting it in the right position to trigger a sound, let alone being able to move all of them to precise enough positions to have a perfectly timed loop. Sometimes the pin flicks also wouldn't be registered, which would put the 'select' object in the Max patch out of sync, and effectively push all of the pins 'forward' one place. Another problem was often after a few 'flicks', an outer pin would be pushed out of position.

Conclusions

The goal of this project was to implement a physical drum machine using a motor from the Elegoo Super Starter Arduino Kit. I particularly enjoyed learning about how to use the Arduino, and wanted to use the knowledge to actually build something I could hold in my hands! As it turned out, the transition from having the idea on paper to sitting in front of me on the desk was riddled with unforeseeable difficulties, but I am very happy to have produced something that spins and produces sound. I also think the idea has significant potential. With a more compact design (perhaps using a flatter base so shorter wires could be used) and stronger materials than hot glue, blu-tack and duct tape, I might have been able to 'slide' the pins and execute the micro-scale timing adjustments I was hoping for. In a commercial device, the circuitry could remain the same, and the max patch could be controlled by some user-friendly interface. The rotation was very satisfying, and I think that a more refined version of a similar drum machine could be a fun addition to studios of musicians who enjoy having an analog-style of control over digital sound generation. Overall, I really enjoyed the creative process and I hope that in the future I'll be able to make prototype number two!