Perpetual Music Machines

"Perpetual motion doesn't exist"

Peter Cancilla
Worcester Polytechnic Institute
pacancilla@wpi.edu

Harrison Rubin
Worcester Polytechnic Institute
hnrubin@wpi.edu

Terence Tan
Worcester Polytechnic Institute
ttan@wpi.edu

ABSTRACT

Perpetual motion machines are intricate machines that create the illusion of infinitely recycled energy. The combination of this machinic concept with sonic production creates an eye-catching performance. This project explores the sonic capabilities of the traditional marble machine but with a twist.

1. CONCEPT, MOTIVATION, AND PURPOSE

Perpetual motion machines are fun to watch because what seems impossible is happening in front of your eyes. This fascination is what motivates us to use this concept in the production of music. The musical machine is intended to incorporate techniques used in perpetual motion machines to energize objects that create sounds. Ultimately, the purpose of this machine is to catch the attention of a person through its visual impossibility and soothing tones."Perpetual motion doesn't exist" will emphasize its form as the visual illusion of perpetual energy is its underlying principle.

2. PRIOR ART

Many inventors have tried to create perpetual motion machines, both practically - attempting to achieve the production of infinite energy - and through illusion. To create the illusion, these devices have hidden power sources that energize certain moving objects. One of the most well known examples is the drinking bird, which creates the illusion of perpetual motion using the temperature difference of water. This works by using a liquid that evaporates at room temperature, usually dichloromethane. As shown in Figure 1, at room temperature, the liquid evaporates and then the vapor pushes the liquid up the tube to a point that it flips over, as shown in Figure 2. The bird's beak would then be in cool water, which cools down the liquid, which then condenses the vapor back into a liquid, and the cycle repeats. The idea of perpetual motion creates mystery and intrigue within the viewer because it seems almost magical, or not able to work, yet it does. This draws visual attention to "Perpetual motion doesn't exist".

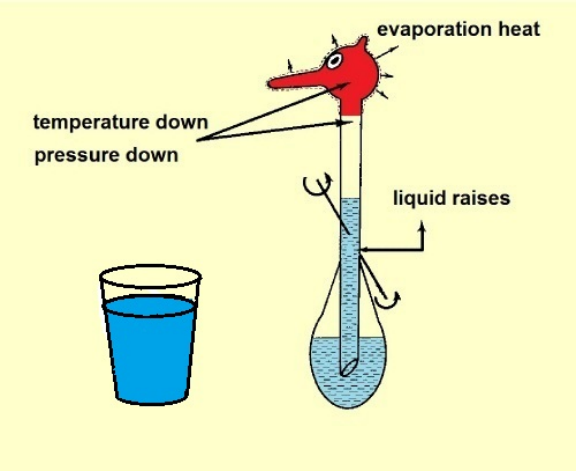

Figure 1. [1]

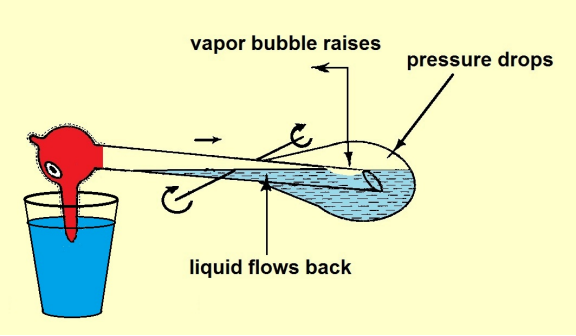

Figure 2. [1]

Another famous example is shown in **Figure 3**, where a hidden electromagnet is placed in the base of the machine, providing kinetic energy to the rolling marble. This neat trick is done by first sensing the marble via an inductive sensor, and an electromagnet to pull the marble through the principles of magnetism. The electromagnet is precisely timed to turn off as soon as the marble passes it such that the magnet doesn't slow down the marble.

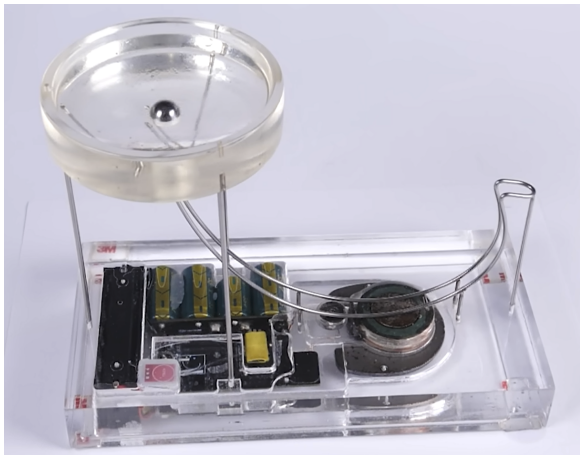

Figure 3: Electromagnet design [2] [3]

Other designs of this device involve a wheel spinning (using a small motor) within the hole at the top, shown in following figures 4 and 5.

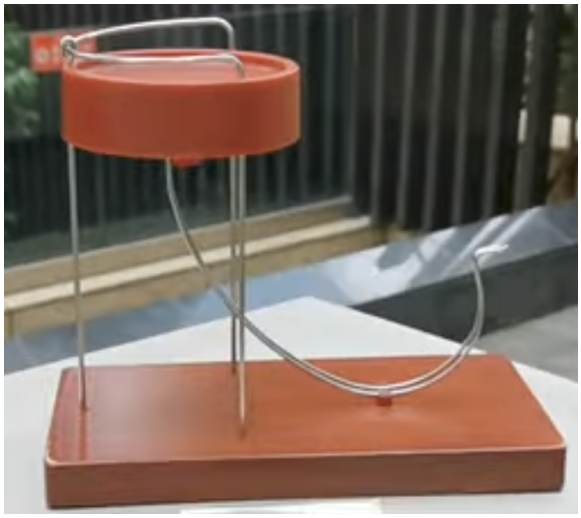

Figure 4: Design using the wheel [4]

Figure 5: The wheel to accelerate the metal ball. [4]

The design with the wheel looks the same barring color and it does succeed in simulating perpetual motion like the electromagnet design, even requiring a less powerful battery: one AA battery (1.5V) vs a 3.7 volt lithium battery. However, the motor that spins the wheel shown is incredibly loud, which would disrupt our musical goals, since the motor is either on or off, it doesn't activate by proximity like the electromagnet design. To generate enough energy to send the ball back up, a battery is used to charge capacitors which release some of the charge collected to an inductive proximity sensor. When power is supplied to the sensor/the coil inside the sensor, an oscillating magnetic field is produced. A changing magnetic field induces a current in nearby metals, in our example it is inducing an oscillating current in the metal ball. This additionally

has the ball produce a magnetic field from that oscillating current. The sensor detects when the total magnetic field decreases, when the ball is over the sensor, then sends an electrical current through an additional wire to activate the electromagnet. The electromagnet and the ball have magnetic fields that repel each other, which is what causes the ball to fly up the curved part of the rail, into the bowl, just to roll down into the hole and start the process again.

3. REQUIREMENTS

The qualities of sound "Perpetual motion doesn't exist" will vary depending on the material/instrument attached to "Perpetual motion doesn't exist". For example a cymbal, a piece of glass/glass container, a piece of wood, a piece of cloth, etc. "Perpetual motion doesn't exist" is essentially a percussion instrument, striking these varying materials/sonic objects with marbles. With this in mind, "Perpetual motion doesn't exist" should be designed and constructed with modularity in mind. The sonic objects should be able to be easily swappable and there should be an opportunity for the number of modules to expand, increasing the dynamic and timbre ranges of "Perpetual motion doesn't exist".

Each module of "Perpetual motion doesn't exist" should be able to operate around 1Hz. The perpetual marble machines that rely on electromagnetic attraction may have brief downtimes to allow for the discharge capacitors to recharge before having enough voltage to accelerate the next marble. However, there must be an acceptable latency for the ball rolling around into the hole, as it is unlikely the ball would fall in the same manner every time. "Perpetual motion doesn't exist" itself is unpitched, tuning and things pertaining to pitch are dependent on what "Perpetual motion doesn't exist" is going to hit. "Perpetual motion doesn't exist" is made to function independently, with no other input than being plugged in to produce acoustic sound.

4. PRELIMINARY DESIGN

The musical machine's Preliminary design **(Figure 7)** is broken down into visual aesthetics, sonic objects, excitation, and structure. Our machines consist of visual and hidden items. Most of the hidden items would be the electronics and microcontrollers that create the illusion of perpetual motion machines. For the visual items, there will be a ramp, basket, and marble.

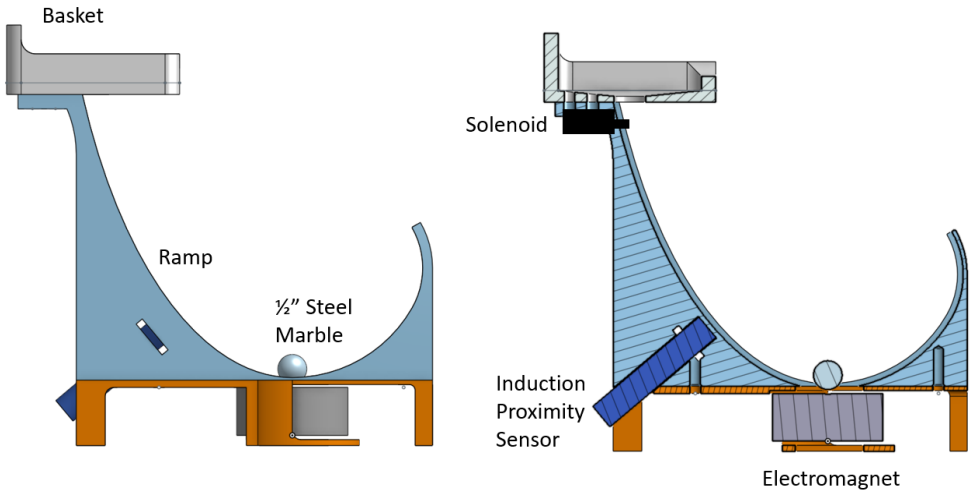

Figure 7. CAD model of Preliminary Design

Shown on the right **(Figure 7)** are the major electrical components that supply energy to the steel marble as it comes down the ramp. The solenoid is able to control whether or not the marble is able to

roll down the ramp. In certain compositions, the solenoid can remain open or closed for a prolonged period of time, or even index marbles down the ramp if there were enough stored in the basket. Once the marble rolls down the ramp and passes the induction proximity sensor, a change in its analog signal will be detected by the microcontroller and turns the electromagnet on for a specified on-time. The magnetic field provided by the electromagnet will propel the steel marble towards it. The on-time will be tuned such that the electromagnet will be turned off before the marble fully passes it.

In this version of the CAD shown above **(Figure 7)**, none of the intended electrical components are hidden from the user because it is a prototype. In the final design, the intention is to fully hide electrical components from the user to best create the visual illusion.

On the back vertical wall of the basket will be our sonic object: glass, metal, and wood as our primary options. These panels of varying materials will be mounted to the basket and will be impacted by the steel marble as it travels back up. The positioning of the panel will avoid bouncing the marble out of the basket.

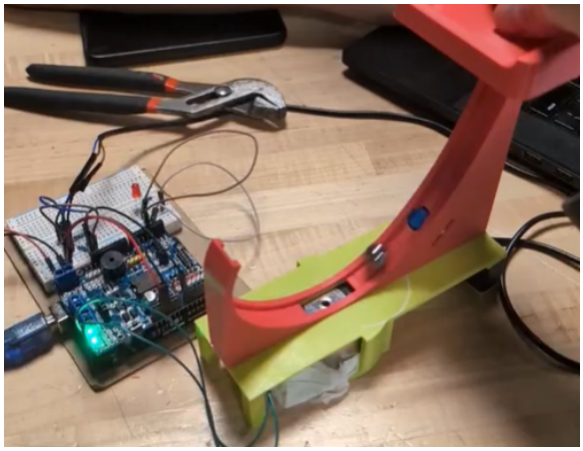

Figure 8. Testing of Preliminary Design

For controlling the indexing solenoid and electromagnet, a 12V 1 amp wall power supply, L298P motor controller shield, and a LJ12A3-4-Z/BX Inductive Sensor were used. This shield can control up to 2 12V components with PWM. Although there is no intention to supply reverse polarity to either components, a h-bridge shield for the arduino uno is well packaged and does not require custom circuitry for a MOSFET.

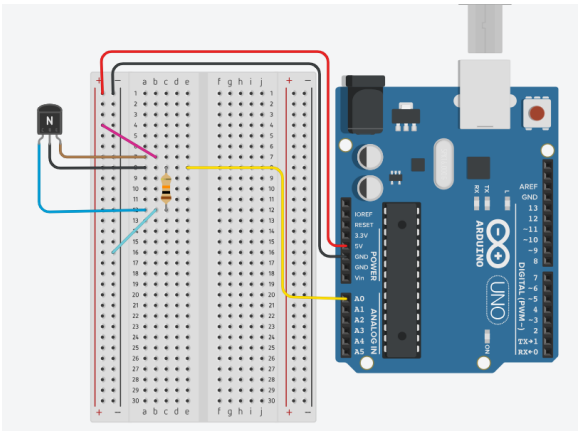

Figure 9. Wiring diagram for inductive sensor

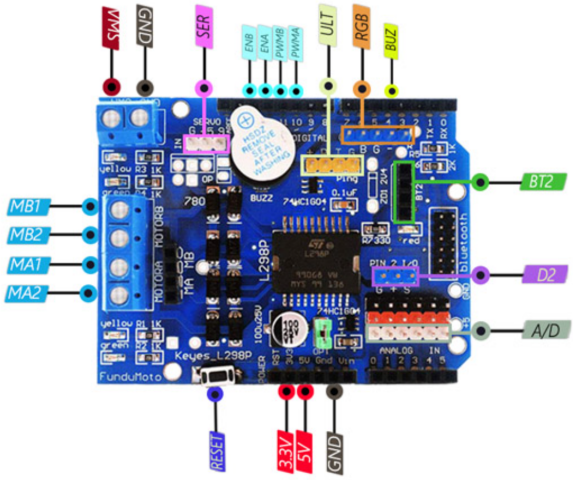

Figure 10. Outputs of L298P Shield [5]

5. FINAL DESIGN

The final design of "Perpetual motion doesn't exist" saw significant alterations due to struggles in getting the electromagnet design to function properly. The new design of "Perpetual motion doesn't exist" unfortunately fails to achieve some of our previously set goals. It requires assistance in the sense that The machine can't reload on its own. The machine doesn't successfully simulate perpetual motion, nor is the way the machine works as concealed and "mysterious" as planned. Although this design does give greater control for composition, keeps most of

the components hidden, is still modular, and as a bonus is interactive.

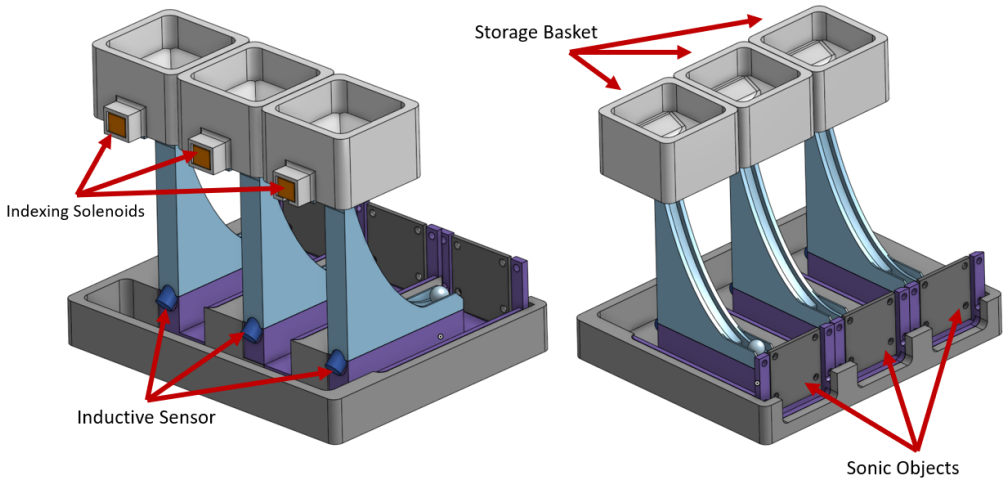

Figure 11. CAD of Final Design

The design retains the solenoids to control whether the marble(s) drop down the ramp. Technically “Perpetual motion doesn’t exist” is composed of three modules that can be removed and replaced. Each module has a sonic object in the shape of a square plate at the bottom of the ramp to be struck by the marble. The inductive sensor allows for the use of virtual sounds. The machine will have two ways of playing/being played when on, “random” and following MIDI messages. It can also be played by loading marbles into the funnels and lightly pulling the solenoids to release a marble/marbles.

In the final design, the main electrical components are the same 12V power supply, LJ12A3-4-Z/BX Inductive Sensor, L298P motor controller shield, and FQP30N06L MOSFET. The motor controller shield only has 2 H-bridges, so a separate MOSFET was used to supply power to the 3rd solenoid.

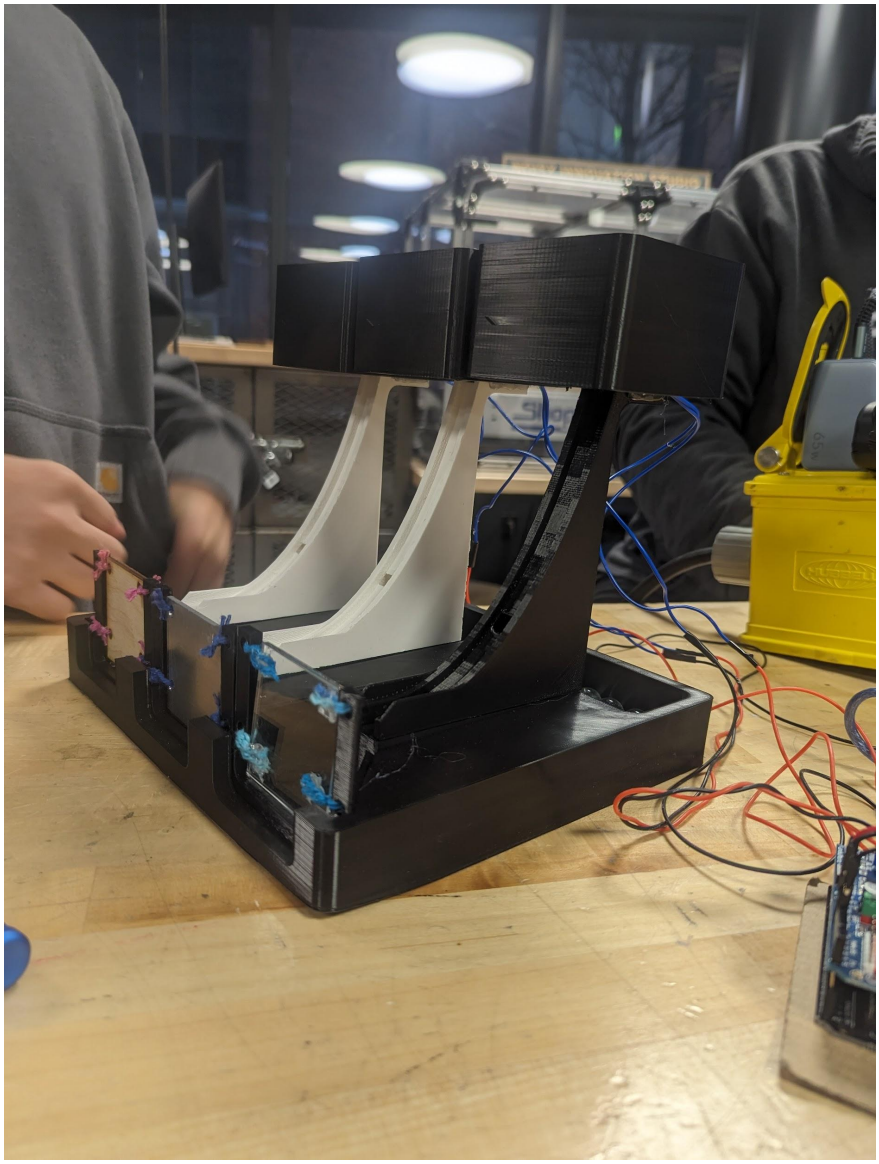

Figure 12. Final Design

6. EVALUATIONS

Throughout testing, we realized that many other parts of the robot were unintentionally making noise. For example, the constant clicking of the solenoid as the shaft slams on both ends of travel significantly dulled out the intentional sounds. A quick fix to this problem was adding compressible padding between the ends of the shaft and solenoid housings. Good options that we found to work were o-ring seals and layers of hot glue.

In addition, the rolling of the marbles along the 3D-printed surfaces was audible as well. To prevent this, most of the surfaces that the marbles will come in contact with were layered with adhesives backed felt.

Since the prototyping phases, jamming of the marbles was a consistent problem that was initially solved in the earlier design when only ferromagnetic marbles were used. We solved the problem by changing the ramp angle of each of the ramps in the bucket. The alteration greatly decreased the chances of multiple marbles trying to enter the ramp at the same time. However, once we changed to glass marbles, we found that the diameter of each

marble differed slightly and was more prone to jamming up at the exit hole.

7. MUSICAL USAGE AND REFLECTION

Reflecting on the initial requirements set at the beginning of the project, only a few were fulfilled. This is mainly a result of our team's inability to get the electromagnet to propel the magnet with enough force to mimic the desired perpetual motion. Due to this, we decided to descope our musical machine and create a product that could achieve as many of the initial requirements as possible. Our final design is a culmination of what we learned through trying to get the electromagnet design to work and a proof of concept that the electromagnet design could have worked. All of the main features included in the initial design besides the electromagnet were featured in the final design. This makes us confident that if we were able to properly incorporate the electromagnet, we would be able to achieve all of our requirements.

The main requirements that we were not able to achieve were the visual appeal of a perpetual machine because the marbles are not automatically loaded back into the storage basket. On the other hand, the modular sonic objects proved to be easily interchangeable and make the soothing sounds that we desired. Overall, the project was majorly successful in proving that the concept of a perpetual motion machine is feasible and within grasp once the kinks of the electromagnet are solved.

For future work on this robot, it would be interesting to see the number of modules be expanded to create a more spatial focused experience. In addition, having the inductive sensors trigger the movement to motors would add another layer to its musical composition. Lastly, pitched percussion could be explored by varying the thickness of each panel to allow for a range of resonant frequencies.

8. REFERENCES

[1] https://www.drinkingbird.eu/howitworks.html
[2]https://www.amazon.com/BMVVQUE-Perpetual-Machine%EF%BC%8CNewest-%EF%BC%8CDecompression-Ornament%EF%BC%8COffice/dp/B0B57FN667/ref=sr_1_8?crid=1GCX4HTPZL6XA&keywords=perpetual%2Bmotion%2Bmachine%2Bmarble&qid=1699563976&sprefix=%2Caps%2C408&sr=8-8&th=1
[3]https://youtu.be/r_LG8FDt51U?si=dlCB1PZKixL-74gg)
[4]https://youtu.be/trC5Dg3Vpi0?si=n8I752Fj-sq7Vp4D)
[5]https://electropeak.com/learn/interfacing-l298p-h-bridge-motor-driver-shield-with-arduino/

9. APPENDIX

Bill of Materials (BOM)		
Item Name	Quantity	Notes
Arduino Uno	1	
L298P Shield	1	
12V Power supply	1	1 amp limit
12V PushPull Solenoid (BM-053OB)	1 per module (3 total)	10mm stroke 5N force 1 amp
Male-Male Jumper Cables	~ 10	

Chromium Steel 0.5” Diameter marbles	10	Must be Ferromagnetic material
Glass 0.5” Diameter	100	
40 kg, 49mm Diameter Lifting electromagnet	1 per module (3 total)	Not used in final design
LJ12A3-4-Z/BX Inductive Sensor	1 per module (3 total)	Detection within 4mm
FQP30N06L MOSFET	1	
In4001 diode	1	
Adhesive Backed Felt	1	

Code for playing using MIDI messages:

```
const byte bufferSize = 2;
byte dataReceived[bufferSize];


int machine_1_pin = 12;
int machine_2_pin = 13;
int machine_3_pin = 5;


int pwm_1 = 10;
int pwm_2 = 11;


int tempo = -1;


//set to high for random
int randomness = 9;
unsigned long msRandom = -1;
int randomThreshold = 30000;
int hyper = false;


void setup() {
 // put your setup code here, to run once:
```

```
 Serial.begin(9600);
 pinMode(machine_1_pin, OUTPUT);
 pinMode(machine_2_pin, OUTPUT);
 pinMode(machine_3_pin, OUTPUT);
 pinMode(pwm_1, OUTPUT);
 pinMode(pwm_2, OUTPUT);
 pinMode(randomness, INPUT);
}


void activateSolenoid(int machineNum) {
 int machine = -1;
 int pwm = -1;
 switch (machineNum) {
   case 1:
     machine = machine_1_pin;
     pwm = pwm_1;
     break;
   case 2:
     machine = machine_2_pin;
     pwm = pwm_2;
     break;
   case 3:
     machine = machine_3_pin;
     pwm = -1;
     break;
 }
 digitalWrite(machine, true);
 if (pwm != -1) {
   digitalWrite(pwm, true);
 }
 delay(50);
 digitalWrite(machine, false);
 if (pwm != -1) {
   digitalWrite(pwm, false);
 }
}

void loop() {
```

```
if (digitalRead(randomness)) {
  if (msRandom == -1) {
    msRandom = millis();
  }
  int solenoidDelay = 500;
  if (!hyper) {
    int timeSince = millis() - msRandom;
    if (timeSince > randomThreshold) {
      hyper = true;
    }
  } else {
    solenoidDelay = 50;
  }
  activateSolenoid(random(3) + 1);
  delay(random(solenoidDelay));
  return;
}

 // put your main code here, to run repeatedly:
if (Serial.available() >= 2) {
  Serial.readBytes(dataReceived, 2);


  int pitch = dataReceived[0];
  Serial.println("recv");
  Serial.println(pitch);
  int velocity = dataReceived[1];
  if (velocity == 0)
    return;


  switch (pitch) {
    case 60:
      activateSolenoid(1);
      break;
    case 61:
      activateSolenoid(2);
      break;
    case 62:
```

```
        activateSolenoid(3);
        break;
      default:
        break;
    }
  }
}
```

Code for playing randomly:

```
const byte bufferSize = 2;
byte dataReceived[bufferSize];


int machine_1_pin = 12;
int machine_2_pin = 13;
int machine_3_pin = 5;


int pwm_1 = 10;
int pwm_2 = 11;


int tempo = -1;


//set to high for random
int randomness = 9;
unsigned long msRandom = -1;
int randomThreshold = 30000;
int hyper = false;


void setup() {
 // put your setup code here, to run once:
 Serial.begin(9600);
 pinMode(machine_1_pin, OUTPUT);
 pinMode(machine_2_pin, OUTPUT);
 pinMode(machine_3_pin, OUTPUT);
 pinMode(pwm_1, OUTPUT);
```

```
 pinMode(pwm_2, OUTPUT);
 pinMode(randomness, INPUT);
}


void activateSolenoid(int machineNum) {
 int machine = -1;
 int pwm = -1;
 switch (machineNum) {
   case 1:
     machine = machine_1_pin;
     pwm = pwm_1;
     break;
   case 2:
     machine = machine_2_pin;
     pwm = pwm_2;
     break;
   case 3:
     machine = machine_3_pin;
     pwm = -1;
     break;
 }
 digitalWrite(machine, true);
 if (pwm != -1) {
   digitalWrite(pwm, true);
 }
 delay(50);
 digitalWrite(machine, false);
 if (pwm != -1) {
   digitalWrite(pwm, false);
 }
}

void loop() {


 if (!digitalRead(randomness)) {
   if (msRandom == -1) {
     msRandom = millis();
   }
```

```
  int solenoidDelay = 500;
  if (!hyper) {
    int timeSince = millis() - msRandom;
    if (timeSince > randomThreshold) {
      hyper = true;
    }
  } else {
    solenoidDelay = 50;
  }
  activateSolenoid(random(3) + 1);
  delay(random(solenoidDelay));
  return;
}

 // put your main code here, to run repeatedly:
if (Serial.available() >= 2) {
  Serial.readBytes(dataReceived, 2);


  int pitch = dataReceived[0];
  Serial.println("recv");
  Serial.println(pitch);
  int velocity = dataReceived[1];
  if (velocity == 0)
    return;


  switch (pitch) {
    case 60:
      activateSolenoid(1);
      break;
    case 61:
      activateSolenoid(2);
      break;
    case 62:
      activateSolenoid(3);
      break;
    default:
      break;
  }
```

```
  }
}
```

The slight difference between this and the Code for using MIDI is an exclamation mark, the not operator, in “if (!digitalRead(randomness))”