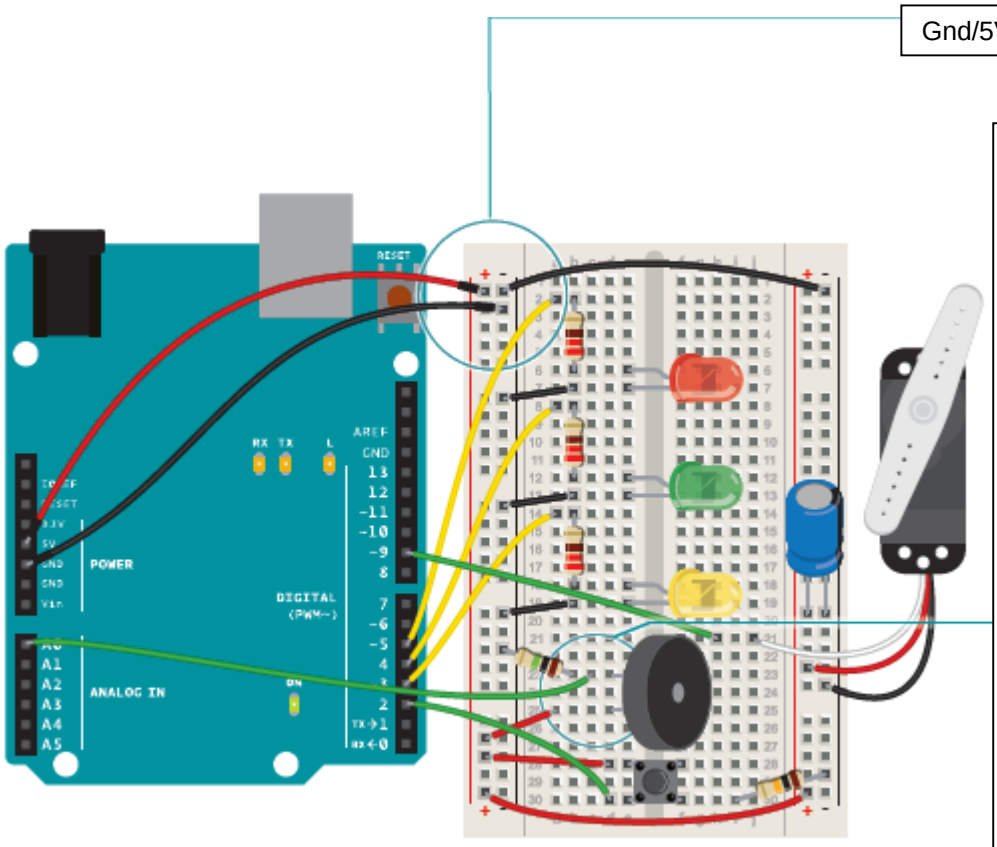
12. Knock lock

The piezo you used for playing back sounds in the theremin and keyboard projects can also be used as an input device. When plugged into 5V, the sensor can detect vibrations that can be read by the Arduino's analog inputs. You'll need to plug in a high value resistor (like 1-megohm) as the reference to ground for this to work well.

When the piezo is pressed flat against a solid surface that can vibrate, like a wooden table top, your Arduino can sense how intense a knock is. Using this information you can check to see if a number of knocks fall in an acceptable range. In code you can track the number of knocks and see if they match your settings.

A switch will let you lock the motor in place. Some LEDs will give you status: a red LED will indicate the box is locked, a green LED will indicate the box is unlocked, and a yellow LED lets you know if a valid knock has been received.

You'll also be writing your own function that will let you know if a knock is too loud or too soft. Writing your own function helps save time programming by reusing code instead of writing it out many times. Functions can take arguments and return values. In this case, you'll give a function the volume of the knock. If it is in the right range, you'll increment a variable.



Place the pushbutton on the breadboard and connect one end to 5V. On the other side of the switch, connect to ground through a 10-kilohm resistor. Connect this junction to digital pin 2 on the Arduino.

Wire up the LEDs, connecting the cathodes (short leg) to ground, and placing a 220-ohm resistor in series with the anodes.

Through their respective resistors, connect the yellow LED to Arduino digital pin 3, the green LED to digital pin 4, and the red LED to digital pin 5.

Insert the male headers into the female socket on the servo motor. Connect the red wire to power, and the black wire to ground. Place a 100uF electrolytic capacitor across power and ground to smooth out any irregularities in voltage, making sure you have the capacitor's polarity correct. Connect the servo's data wire to pin 9 on your Arduino

Attach the wires from the piezo to the breadboard. Attach one wire to power. If your piezo has a red wire or one marked with a "+", that is the one to connect to power. If your piezo doesn't indicate polarity, then you can hook it up either way. Wire the other end of the piezo to Analog Pin 0 on your Arduino. Place a 1-megohm resistor between the ground and the other wire. Lower resistor values will make the piezo less sensitive to vibrations.

```
// import the library
#include <Servo.h>
// create an instance of the servo library
Servo myServo;

const int piezo = A0;      // pin the piezo is attached to
const int switchPin = 2;    // pin the switch is attached to
const int yellowLed = 3;    // pin the yellow LED is attached to
const int greenLed = 4;    // pin the green LED is attached to
const int redLed = 5;   // pin the red LED is attached to

// variable for the piezo value
int knockVal;
// variable for the switch value
int switchVal;

// variables for the high and low limits of the knock value
const int quietKnock = 10;
const int loudKnock = 100;

// variable to indicate if locked or not
boolean locked = false;
// how many valid knocks you've received
int numberOfKnocks = 0;

void setup() {
  // attach the servo to pin 9
  myServo.attach(9);

  // make the LED pins outputs
  pinMode(yellowLed, OUTPUT);
  pinMode(redLed, OUTPUT);
  pinMode(greenLed, OUTPUT);

  // set the switch pin as an input
  pinMode(switchPin, INPUT);

  // start serial communication for debugging
  Serial.begin(9600);

  // turn the green LED on
  digitalWrite(greenLed, HIGH);

  // move the servo to the unlocked position
  myServo.write(0);

  // print status to the serial monitor
  Serial.println("the box is unlocked!");
}

void loop() {

  // if the box is unlocked
  if (locked == false) {
```

```
  switchVal = digitalRead(switchPin);

  // if the button is pressed, lock the box
  if (switchVal == HIGH) {
    // set the locked variable to "true"
    locked = true;

    // change the status LEDs
    digitalWrite(greenLed, LOW);
    digitalWrite(redLed, HIGH);

    // move the servo to the locked position
    myServo.write(90);

    // print out status
    Serial.println("the box is locked!");

    // wait for the servo to move into position
    delay(1000);
  }
}

// if the box is locked
if (locked == true) {

  // check the value of the piezo
  knockVal = analogRead(piezo);

 // if there are not enough valid knocks
 if (numberOfKnocks < 3 && knockVal > 0) {

   // check to see if the knock is in range
   if (checkForKnock(knockVal) == true) {

     // increment the number of valid knocks
     numberOfKnocks++;
   }

   // print status of knocks
   Serial.print(3 - numberOfKnocks);
   Serial.println(" more knocks to go");
 }

 // if there are three knocks
 if (numberOfKnocks >= 3) {
   // unlock the box
   locked = false;

   // move the servo to the unlocked position
   myServo.write(0);

   // wait for it to move
   delay(20);

   // change status LEDs
```

```
      digitalWrite(greenLed, HIGH);
      digitalWrite(redLed, LOW);
      Serial.println("the box is unlocked!");

      numberOfKnocks = 0;
    }
  }
}

// this function checks to see if a
// detected knock is within max and min range
boolean checkForKnock(int value) {
  // if the value of the knock is greater than
  // the minimum, and larger than the maximum
  if (value > quietKnock && value < loudKnock) {
    // turn the status LED on
    digitalWrite(yellowLed, HIGH);
    delay(50);
    digitalWrite(yellowLed, LOW);
    // print out the status
    Serial.print("Valid knock of value ");
    Serial.println(value);
    // return true
    return true;
  }
  // if the knock is not within range
  else {
    // print status
    Serial.print("Bad knock value ");
    Serial.println(value);
    // return false
    return false;
  }
}
```

You'll need to import the Servo library and create an instance to use the motor.
Create constants to name your inputs and outputs. Create variables to hold the values from your switch and piezo. Set up some constants to use as thresholds for the knock maximum and minimum levels.
The locked variable will let you know if the lock is enganged or not. A boolean is a data type that can only be true (1) or false (0). You should start with the mechanism unlocked.
The last global variable will hold the number of valid knocks you have received.
In your setup(), attach the servo to pin 9.
Initialize serial communication with the computer so you can monitor the knock volume, what the current state of the lock is, and how many more knocks you have to go.
Turn on the green LED, move the servo to the unlocked position, and print the current status to the serial monitor indicating the circuit is in the unlocked position.
In the loop(), you'll first check to see if the box is locked or not.
This will determine what happens in the rest of the program. If it is locked, read the switch value.
If the switch is closed (you're pressing it), change the locked variable to true, indicating the lock is engaged. Turn the green LED off, and the red LED on. If you don't have the serial monitor on, this is helpful visual feedback to let you know the status of
the lock. Move the servo into the lock position, and print out a message to the serial monitor

indicating the box is now locked.
Add a delay so the lock has plenty of time to move into place. If the locked variable is true, and the lock is engaged, read the value of the vibration of the piezo and store it in knockVal. The next statement checks to see if you have fewer than three valid knocks, and there is some vibration on the sensor. If these are both true, check to see if this current knock is valid or not and increment the numberOfKnocks variable. This is where you'll call your custom function checkForKnocks(). You'll write the function once you're finished with the loop(), but you already know you're going to be asking it if this is a valid knock, so pass the knockVal along as an argument. After checking your function, print out the number of knock still needed.
Check to see if you have three or more valid knocks. If this is true, change the locked variable to false, and move the servo to the unlocked position. Wait for a few milliseconds to let it start moving, and change the status of the green and red LEDs. Print out a status message to the serial monitor, indicating the box is unlocked.
Close up the else statement and the loop() with a pair of curly brackets.
Now it's time to write the function checkForKnock(). When you're writing functions of your own, you need to indicate if it is going to return a value or not. If it is not going to return a value, you declare it as type void, similar to the loop() and setup() functions. If it is going to return a value, you must declare what kind (int, long, float, etc.). In this case, you're checking to see if a knock is valid (true) or not (false). Declare the function as type boolean.
This particular function will be checking a number (your variable knockVal) to see if it is valid or not. To pass this variable along to the function, you create a named parameter when you declare the function.
In your function, whenever you refer to value it will use whatever number it receives as an argument in the main program. At this point value will be set to whatever knockVal is.
Check to see if value is greater than your quiet knock, and less than your loud knock.
If the value falls between those two values it's a valid knock. Blink the yellow LED once and print the value of the knock to the serial monitor.
To let the main program know what the outcome of the comparison is, you use the command return. You use the return command, which also terminates the function: once it executes, you return to the main program.
If value is either too quiet or too loud, print it out to the serial monitor and return false.
Close up your function with one more bracket.