6.Light theremin

A theremin is an instrument that makes sounds based on the movements of a musician's hands around the instrument. You've probably heard one in scary movies. The theremin detects where a performer's hands are in relation to two antennas by reading the capacitive change on the antennas. These antennas are connected to analog circuitry that create the sound. One antenna controls the frequency of the sound and the other controls volume. While the Arduino can't exactly replicate the mysterious sounds from this instrument, it is possible to emulate them using the **tone()** function. Fig. 1 shows the difference between the pulses emitted by **analogWrite()** and **tone().** This enables a transducer like a speaker or piezo to move back and forth at different speeds.

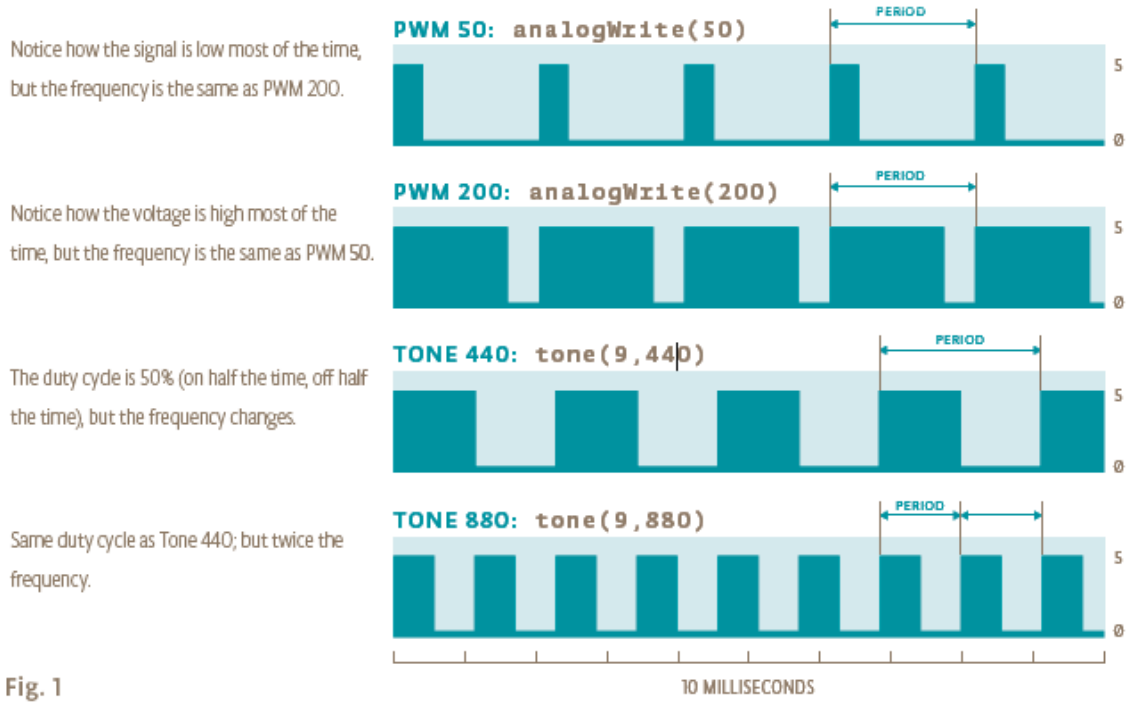
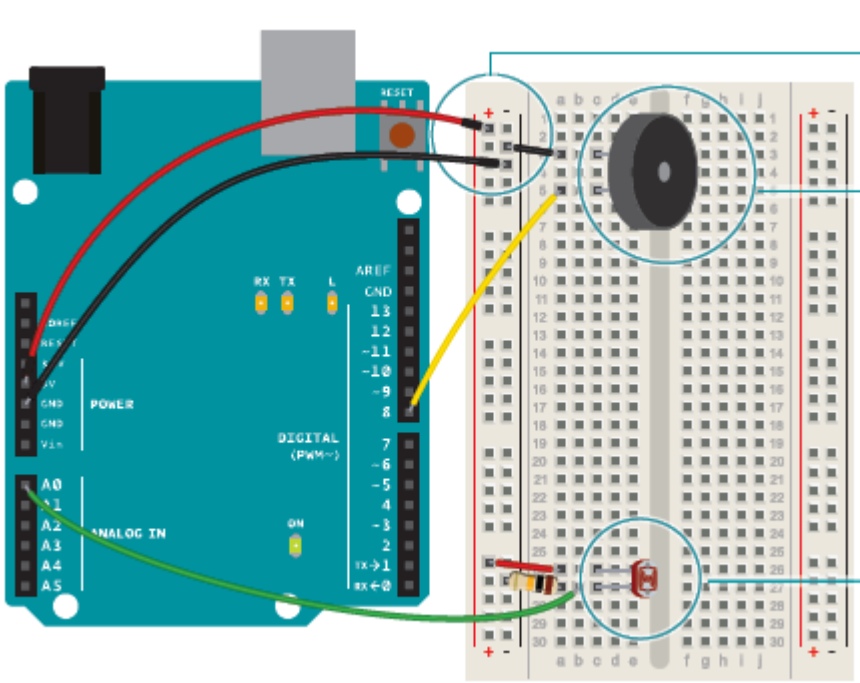
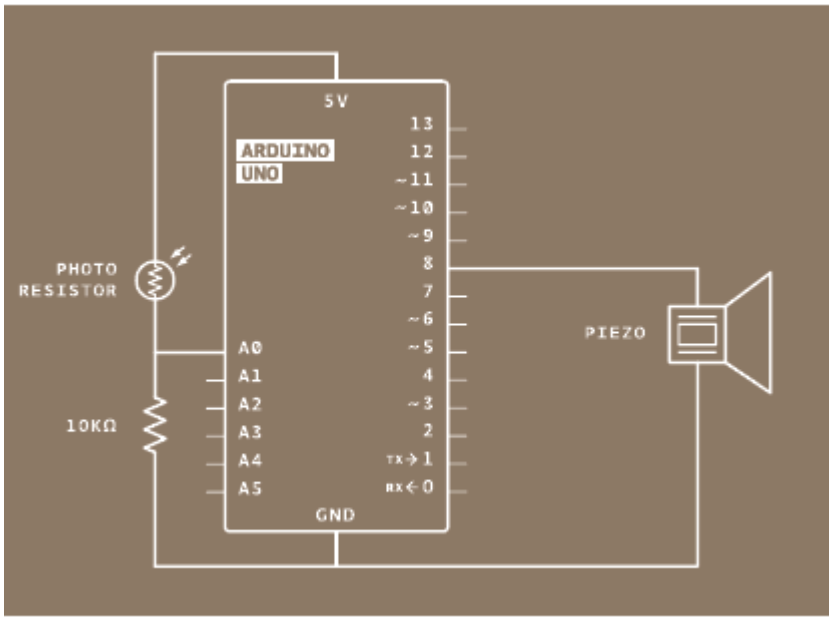


Fig. 1

Instead of sensing capacitance with the Arduino, you'll be using a photoresistor to detect the amount of light. By moving your hands over the sensor, you'll change the amount of light that falls on the photoresistor's face, as you did in Project 4. The change in the voltage on the analog pin will determine what frequency note to play.

You'll connect the photoresistors to the Arduino using a voltage divider circuit like you did in Project 4. You probably noticed in the earlier project that when you read this circuit using analogRead(), your readings didn't range all the way from 0 to 1023. The fixed resistor connecting to ground limits the low end of the range, and the brightness of your light limits the high end. Instead of settling for a limited range, you'll calibrate the sensor readings getting the high and low values, mapping them to sound frequencies using the **map()** function to get as much range out of your theremin as possible. This will have the added benefit of adjusting the sensor readings whenever you move your circuit to a new environment, like a room with different light conditions.


Gnd-5V
Take your piezo, and connect one end to ground, and the other to digital pin 8 on the Arduino.
Place your photoresistor on the breadboard, connecting one end to 5V. Connect the other end to the Arduino's analog In pin 0, and to ground through a 10-kilohm resistor. This circuit is the same as the voltage divider circuit in Project 4.


5V
ARDUINO
UNO
PHOTO
RESISTOR
10KΩ
PIEZO
A0
A1
A2
A3
A4
A5
13
12
~11
~10
~9
8
7
~6
~5
4
~3
2
TX→1
RX←0
GND

```
// variable to hold sensor value
int sensorValue;
// variable to calibrate low value
int sensorLow = 1023;
// variable to calibrate high value
int sensorHigh = 0;
// LED pin
const int ledPin = 13;

void setup() {
  // Make the LED pin an output and turn it on
  pinMode(ledPin, OUTPUT);
  digitalWrite(ledPin, HIGH);

  // calibrate for the first five seconds after program runs
  while (millis() < 5000) {
    // record the maximum sensor value
    sensorValue = analogRead(A0);
    if (sensorValue > sensorHigh) {
      sensorHigh = sensorValue;
    }
    // record the minimum sensor value
    if (sensorValue < sensorLow) {
      sensorLow = sensorValue;
    }
  }
  // turn the LED off, signaling the end of the calibration period
  digitalWrite(ledPin, LOW);
}

void loop() {
  //read the input from A0 and store it in a variable
  sensorValue = analogRead(A0);

  // map the sensor values to a wide range of pitches
  int pitch = map(sensorValue, sensorLow, sensorHigh, 50, 4000);

  // play the tone for 20 ms on pin 8
  tone(8, pitch, 20);

  // wait for a moment
  delay(10);
}
```

Create a variable to hold the analogRead() value from the photoresistor. Next, create variables for the high and low values. You're going to set the initial value in the sensorLow variable to 1023, and set the value of the sensorHigh variable to 0. When you first run the program, you'll compare these numbers to the sensor's readings to find the real maximum and minimum values. Create a constant named ledPin. You'll use this as an indicator that your sensor has finished calibrating. For this project, use the **on-board LED connected to pin 13.**

In the setup(), change the pinMode() of ledPin to OUTPUT, and turn the light on.

The next steps will calibrate the sensor's maximum and minimum values. You'll use a **while()** statement to run a **loop for 5 seconds**. while() loops run until a certain condition is met. In this case you're going to use the **millis()** function to check the current time. millis() reports how long the Arduino has been running since it was last powered on or reset.
In the loop, you'll read the value of the sensor; if the value is less than sensorLow (initially 1023), you'll update that variable. If it is greater than sensorHigh (initially 0), that gets updated. When 5 seconds have passed, the while() loop will end. Turn off the LED attached to pin 13. You'll use the sensor high and low values just recorded to scale the frequency in the main part of your program.
In the loop(), read the value on A0 and store it in sensorValue. Create a variable named pitch. The value of pitch is going to be mapped from sensorValue. Use sensorLow and sensorHigh as the bounds for the incoming values. For starting values for output, try 50 to 4000. These numbers set the range of frequencies the Arduino will generate.
Next, call the **tone()** function to play a sound. It takes three arguments: what pin to play the sound on (in this case pin 8), what frequency to play (determined by the pitch variable), and how long to play the note (try 20 milliseconds to start).
Then, call a delay() for 10 milliseconds to give the sound some time to play.

When you first power the Arduino on, there is a 5 second window for you to calibrate the sensor. To do this, move your hand up and down over the photoresistor, changing the amount of light that reaches it. The closer you replicate the motions you expect to use while playing the instrument, the better the calibration will be.
After 5 seconds, the calibration will be complete, and the LED on the Arduino will turn off. When this happens, you should hear some noise coming from the piezo! As the amount of light that falls on the sensor changes, so should the frequency that the piezo plays.