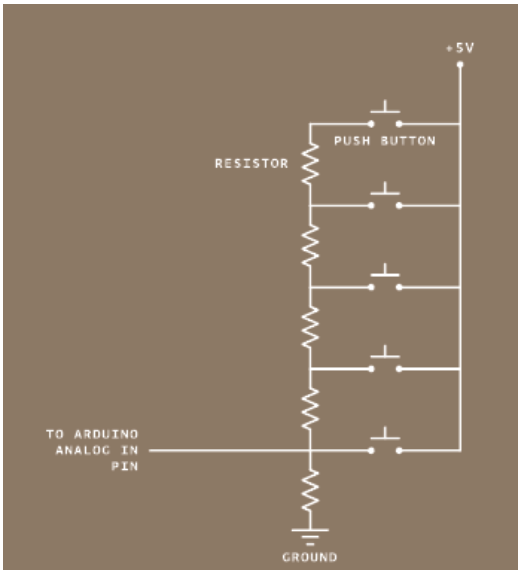
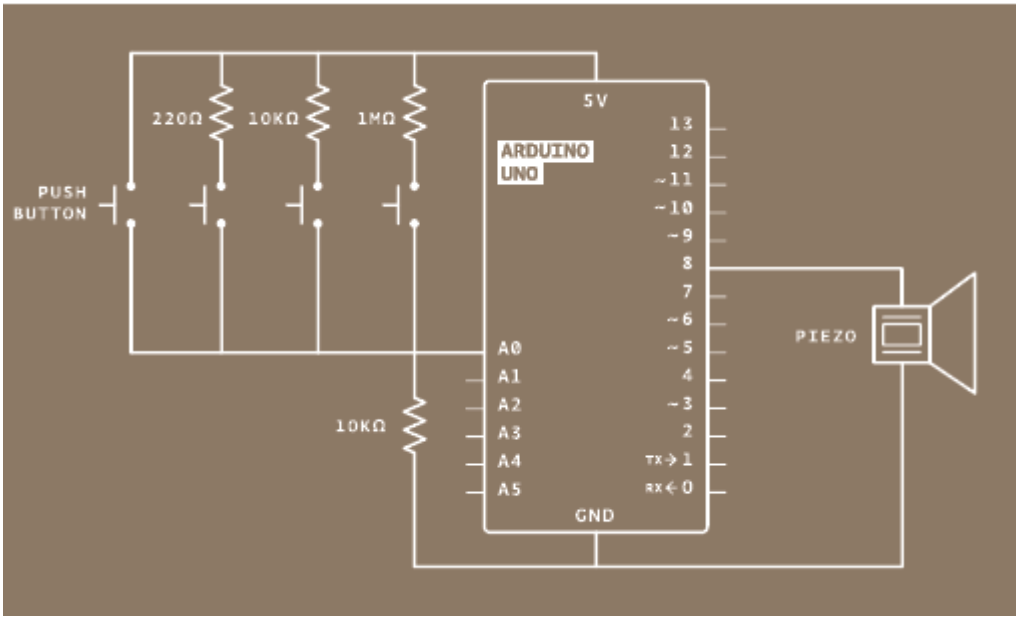
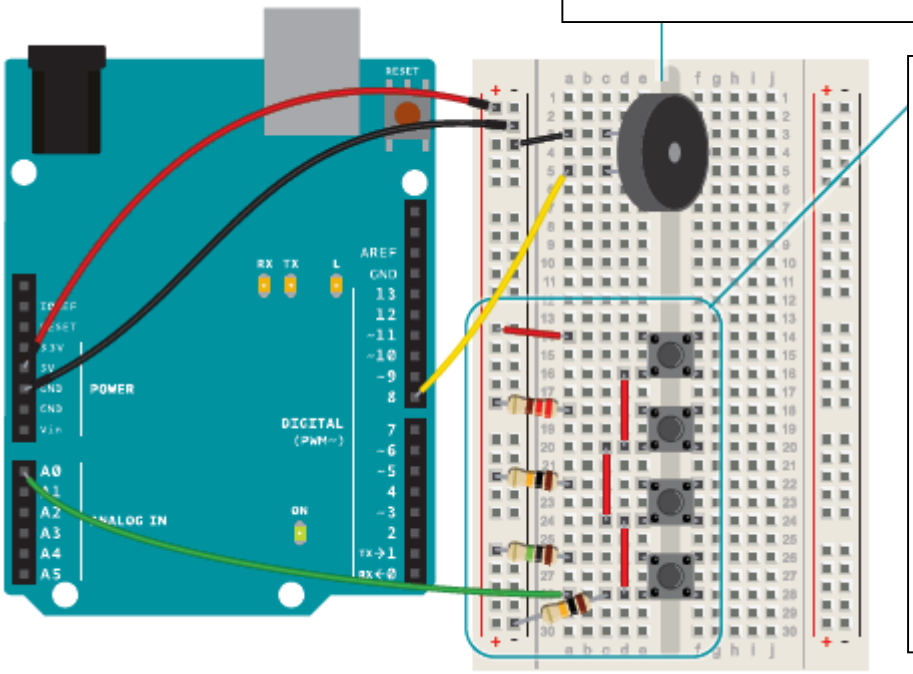
7.keyboard instrument

In this project, you'll be constructing a **resistor ladder.**
This is a way to read a number of switches using the analog input. It's a helpful technique if you find yourself short on digital inputs. You'll hook up a number of switches that are connected in parallel to analog in 0. Most of these will connect to power through a resistor. When you press each button, a different voltage level will pass to the input pin. If you press two buttons at the same time, you'll get a unique input based on the relationship between the two resistors in parallel.







In this program, you'll need to keep a list of frequencies you want to play when you press each of your buttons. You can start out with the frequencies for middle C, D, E and F (262Hz, 294Hz, 330Hz, and 349Hz). To do this, you'll need a new kind of variable called an array.
To declare an array, start as you would with a variable, but follow the name with a pair of square brackets: []. After the equals sign, you'll place your elements in curly brackets.
To read or change the elements of the array, you reference the individual element using the array name and then the index of the item you want to address. The index refers to the order in which the items appear when the array is created. The first item in the array is item 0, the second is item 1, and so forth.

```
// create an array of notes
// the numbers below correspond to the frequencies of middle C, D, E, and F
int notes[] = {262, 294, 330, 349};

void setup() {
  //start serial communication
  Serial.begin(9600);
}

void loop() {
  // create a local variable to hold the input on pin A0
  int keyVal = analogRead(A0);
  // send the value from A0 to the Serial Monitor
  Serial.println(keyVal);

  // play the note corresponding to each value on A0
  if (keyVal == 1023) {
    // play the first frequency in the array on pin 8
    tone(8, notes[0]);
  } else if (keyVal >= 990 && keyVal <= 1010) {
    // play the second frequency in the array on pin 8
    tone(8, notes[1]);
  } else if (keyVal >= 505 && keyVal <= 515) {
    // play the third frequency in the array on pin 8
    tone(8, notes[2]);
  } else if (keyVal >= 5 && keyVal <= 10) {
    // play the fourth frequency in the array on pin 8
    tone(8, notes[3]);
  } else {
    // if the value is out of range, play no tone
    noTone(8);
  }
}
```

In your setup(), start serial communication with the computer. In the loop(), declare a local variable to hold the value read on pin A0. Because each switch has a different resistor value connecting it to power, each will have a different value associated with it. To see the values, add the line **Serial.println(keyVal)** to send to the computer.

Using an if()...else statement, you can assign each value to a different tone. The values included in the example program are ballpark figures for these resistor sizes. **As all resistors have some tolerance for error, these may not work exactly for you. Use the information from the serial monitor to adjust as necessary.**

After each if() statement, call the tone() function. The program references the array to determine what frequency to play. If the value of A0 matches one of your if statements, you can tell the Arduino to play a tone. It's possible your circuit is a little "noisy" and the values may fluctuate a little bit while pressing a switch. To accommodate for this variation, it's a good idea to have a small range of values to check against. If you use the comparison "&&", you can check multiple statements to see if they are true.

If you press the first button, notes[0] will play. If you press the second, notes[1] will play, and if you press the third, notes[2] will play. Only one frequency can play on a pin at any given time, so if you're pressing multiple keys, you'll only hear one sound.

To stop playing notes when there is no button being pressed, call the noTone() function, providing the pin number to stop playing sound on.