Simple Piezoelectric Speaker

Source (ECE-Depot): http://www.cui.com/product/resource/cvs-2308.pdf

This simple speaker is fairly quiet but can be operated directly off any PWM pin without the need for a driver circuit.

Music on a Microcontroller

Since a microcontroller has no tuned elements (like a string or pipe of the correct length), we must specify the notes to be played by their frequency. A lookup table of the appropriate frequency for the notes on a piano is available on Wikipedia (https://en.wikipedia.org/wiki/Piano_key_frequencies).

The “tone()” command is installed by default in the Particle IDE, thus no #include statements are required. The tone() command generates a square wave of the specified frequency and duration (and 50% duty cycle) on a timer channel pin (D0, D1, A0, A1, A4, A5, A6, A7, RX, TX). Use of the tone() function will interfere with PWM output on the selected pin.

tone(Pin,Frequency,Duration) takes three arguments:

- Pin - the pin on which to generate the tone (D0)
- Frequency - the frequency of the tone in hertz (440)
- Duration - the duration of the tone in milliseconds (1000)
    - A zero value makes the tone continuous

Documentation is available at:
https://docs.particle.io/reference/firmware/photon/#tone-

Minimal Working Example

This example creates a two tone siren that plays for 4 cycles.

Wiring Diagram

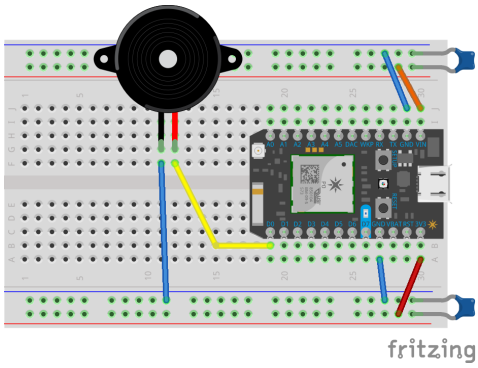

Figure 1: Example Circuit Wiring Diagram

Code

```
int speakerPin = D0;

void setup() {
  // repeat the siren 4 times
  for (int i = 0; i < 4; i++) {
    tone(speakerPin, 880, 1000);
    delay(1100);
    tone(speakerPin, 440, 1000);
    delay(1100);
  }
}

void loop() {
    noTone(speakerPin);
}
```

A more interesting example - A song!

This example will play a few notes from the Zelda theme.

It is worth noting that in the code below “durations” are actually the inversion of the length of a note. In this case if a whole note is 1 second long, then a “duration” 8 note is the length of an 1/8th note, or 0.125 seconds in length. A duration 4 note is the length of a quarter note, or 0.25 seconds in length.

Code

```
// Plays a melody - Connect small speaker to pin D0

int speakerPin = D0;

// notes in the melody:
int melody[] = {293.665,
0,
0,
293.665,
311.127,
349.228,
369.994,
415.305,
0,
415.305,
415.305,
466.164,
523.251,
554.365,
698.456,
659.255,
0,
523.251};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {16,
16,
16,
16,
16,
16,
2,
8,
8,
16,
```

```
16,
16,
1.333333333,
4,
8,
8,
2};

void setup() {
  // iterate over the notes of the melody:
  for (int thisNote = 0; thisNote < arraySize(melody); thisNote++) {

    // to calculate the note duration, take one second
    // divided by the note type.
    //e.g. quarter note = 1500 / 4, eighth note = 1000/8, etc.
    int noteDuration = 1500/noteDurations[thisNote];
    tone(speakerPin, melody[thisNote],noteDuration);

    // to distinguish the notes, set a min time between them.
    // the note's duration + 30% seems to work well:
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    // stop the tone playing:
    noTone(speakerPin);
  }
}
```

Keep in mind this method of making “music” is **very limited**. This method can only play one note/tone at a time (ie., no melody + harmony). Also, the harmonics of these note are fixed. (ie., you can’t make it sound like a trumpet or a piano...it’s just a harsh metallic square wave at a particular frequency.) It is much more complex to play multiple simultaneous tones and even more complex again to play full “waveform” data. For those that are really interested, these things **are** possible with the Photon, but will require significant independent research and learning.