

Project 12



Piezo Sounder Melody Player

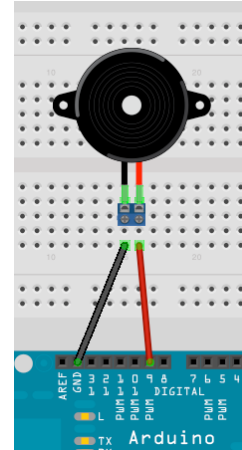
Project 12 – Piezo Sounder Melody Player

In this project we are going to use a super simple circuit to produce sounds from our Arduino using a Piezo Sounder.

Connect it up

What you will need

| | |
|----------------|---|
| Piezo Disc |  |
| Terminal Block |  |



(courtesy of <http://www.arduino.cc/en/Tutorial/Melody>)

```
// Project 12 – Melody Player
int speakerPin = 9;
int length = 15; // the number of notes
char notes[] = "ccggaagffeeddc "; // a space represents a rest
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 };
int tempo = 300;

void playTone(int tone, int duration) {
  for (long i = 0; i < duration * 1000L; i += tone * 2) {
    digitalWrite(speakerPin, HIGH);
    delayMicroseconds(tone);
    digitalWrite(speakerPin, LOW);
    delayMicroseconds(tone);
  }
}

void playNote(char note, int duration) {
  char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
  int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 };
  // play the tone corresponding to the note name
  for (int i = 0; i < 8; i++) {
    if (names[i] == note) {
      playTone(tones[i], duration);
    }
  }
}

void setup() {
  pinMode(speakerPin, OUTPUT);
}

void loop() {
  for (int i = 0; i < length; i++) {
    if (notes[i] == ' ') {
      delay(beats[i] * tempo); // rest
    } else {
      playNote(notes[i], beats[i] * tempo);
    }

    // pause between notes
    delay(tempo / 2);
  }
}
```

When you run this code the Arduino will play a very nice (yeah ok it's terrible) rendition of 'Twinkle Twinkle Little Star'. Sounding very similar to those annoying birthday cards you can buy that play a tune when you open it up.

Let's take a look at this code and see how it works and find out what a piezo disc is.

Project 12 – Code Overview

In this project we are making sounds using a piezo disc. A piezo disc can do nothing more than make a click when we apply a voltage to it. So to get the tones we can hear out of it we need to make it click many times a second fast enough that it becomes a recognisable note.

The program starts off by setting up the variables we need. The piezo sounders positive (red) cable is attached to Pin 9.

```
int speakerPin = 9;
```

The tune we are going to play is made up of 15 notes.

```
int length = 15; // the number of notes
```

The notes of the tune are stored in a character array as a text string.

```
char notes[] = "ccggaagffeeddc ";
```

Another array, this time of integers, is set up to store the length of each note.

```
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 };
```

And finally we set a tempo for the tune to be played at,

```
int tempo = 300;
```

Next you will notice that we declare two functions before our setup() and loop() functions. It doesn't matter if we put our own functions before or after setup() and loop(). When the program runs, the code within these two functions will not run before setup() runs as we have not called those functions yet.

Let's look at the setup and loop functions before we look at the playTone and playNote functions.

All that happens in setup() is we assign the speaker pin (9) as an output.

```
void setup() {
  pinMode(speakerPin, OUTPUT);
}
```

In the main program loop we have an if/else statement inside a for loop.

```
for (int i = 0; i < length; i++) {
  if (notes[i] == ' ') {
    delay(beats[i] * tempo); // rest
  } else {
    playNote(notes[i], beats[i] * tempo);
  }
}
```

As you can see, the first if statement has as its condition, that the array element [i] that the element contains a space character.

```
if (notes[i] == ' ')
```

If this is TRUE then the code within its block is executed.

```
delay(beats[i] * tempo); // rest
```

and this simply works out the value of beats[i] * tempo and causes a delay of that length to cause a rest in the notes. We then have an else statement.

```
else {
  playNote(notes[i], beats[i] * tempo);
}
```

After an if statement we can extend it with an else statement. An else statement is carried out if the condition within the if statement is false. So, for example. Let's say we had an integer called test and its value was 10 and this if/else statement:

```
if (test == 10) {
  digitalWrite(ledPin, HIGH)
} else {
  digitalWrite(ledPin, LOW)
}
```

Then if 'test' had a value of 10 (which it does) the ledPin would be set to HIGH. If the value of test was anything other than 10, the code within the else statement would be carried out instead and the ledPin would be set to LOW.

The else statement calls a function called playNote and passes two parameters. The first parameter is the value of notes[i] and the second is the value calculated from beats[i] * tempo.

```
playNote(notes[i], beats[i] * tempo);
```

After if/else statement has been carried out, there is a delay whose value is calculated by dividing tempo by 2.

```
delay(tempo / 2);
```

Let us now take a look at the two functions we have created for this project.

The first function that is called from the main program loop is playNote.

```
void playNote(char note, int duration) {
  char names[] = { 'c', 'd', 'e', 'f', 'g', 'a',
    'b', 'C' };
  int tones[] = { 1915, 1700, 1519, 1432, 1275,
    1136, 1014, 956 };
  // play the tone corresponding to the note name
  for (int i = 0; i < 8; i++) {
    if (names[i] == note) {
      playTone(tones[i], duration);
    }
  }
}
```

Two parameters have been passed to the function and within the function these have been given the names note (character) and duration (integer).

The function sets up a local variable array of data type char called 'names'. This variable has local scope so is only visible to this function and not outside of it.

This array stores the names of the notes from middle C to high C.

We then create another array of data type integer and this array stores numbers that correspond to the frequency of the tones, in Kilohertz, of each of the notes in the names[] array.

```
int tones[] = { 1915, 1700, 1519, 1432, 1275,
  1136, 1014, 956 };
```

After setting up the two arrays there is a for loop that looks through the 8 notes in the names[] array and compares it to the note sent to the function.

```
for (int i = 0; i < 8; i++) {
  if (names[i] == note) {
    playTone(tones[i], duration);
  }
}
```

The tune that is sent to this function is 'ccggaagffeeddc' so the first note will be a middle C. The for loop compares that note with the notes in the names[] array and if there is a match, calls up the second function, called playTone, to play the

corresponding tone using in the tones[] array using a note length of 'duration'.

The second function is called playTone.

```
void playTone(int tone, int duration) {
  for (long i = 0; i < duration * 1000L; i +=
    tone * 2) {
    digitalWrite(speakerPin, HIGH);
    delayMicroseconds(tone);
    digitalWrite(speakerPin, LOW);
    delayMicroseconds(tone);
  }
}
```

Two parameters are passed to this function. The first is the tone (in kilohertz) that we want the piezo speaker to reproduce and the second is the duration (made up by calculating beats[i] * tempo).

The function starts a for loop

```
for (long i = 0; i < duration * 1000L; i += tone
  * 2)
```

As each for loop must be of a different length to make each note the same length (as the delay differs between clicks to produce the desired frequency) the for loop will run to 'duration' multiplied by 1000 and the increment of the loop is the value of 'tone' multiplied by 2.

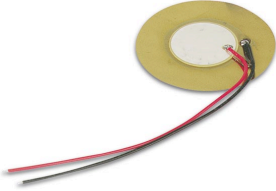
Inside the for loop we simply make the pin connected to the piezo speaker go high, wait a short period of time, then go low, then wait another short period of time, then repeat.

```
digitalWrite(speakerPin, HIGH);
delayMicroseconds(tone);
digitalWrite(speakerPin, LOW);
delayMicroseconds(tone);
```

These repetitive clicks, of different lengths and with different pauses (of only microseconds in length) in between clicks, makes the piezo produce a tone of varying frequencies.

Project 12 – Hardware Overview

The only piece of hardware used in this project is a piezo sounder. This simple device is made up of a thin layer of ceramic bonded to a metallic disc.



Piezoelectric materials, which are some crystals and ceramics, have the ability to produce electricity when mechanical stress is applied to them. The effect finds useful applications such as the production and detection of sound, generation of high voltages, electronic frequency generation, microbalances, and ultra fine focusing of optical assemblies.

The effect is also reversible, in that if an electric field is applied across the piezoelectric material it will cause the material to change shape (by as much as 0.1% in some cases).

To produce sounds from a piezo disc, an electric field is turned on and off very fast, to make the material change shape and hence cause a 'click' as the disc pops out and back in again (like a tiny drum). By changing the frequency of the pulses, the disc will deform hundreds or thousands of times per second and hence causing the buzzing sound. By changing the frequency of the clicks and the time in between them, specific notes can be produced.

You can also use the piezo's ability to produce an electric field to measure movement or vibrations.

Exercise

1. Change the notes and beats to make other tunes such as 'Happy Birthday' or 'Merry Christmas'.
2. Write a program to make a rising and falling tone from the piezo, similar to a car alarm or police siren.