# Arduino Lab 2 - Creating sound and music

In this lab you will be learning how to use the buzzer module, then you will be learning how to play a melody from a song of your choosing.

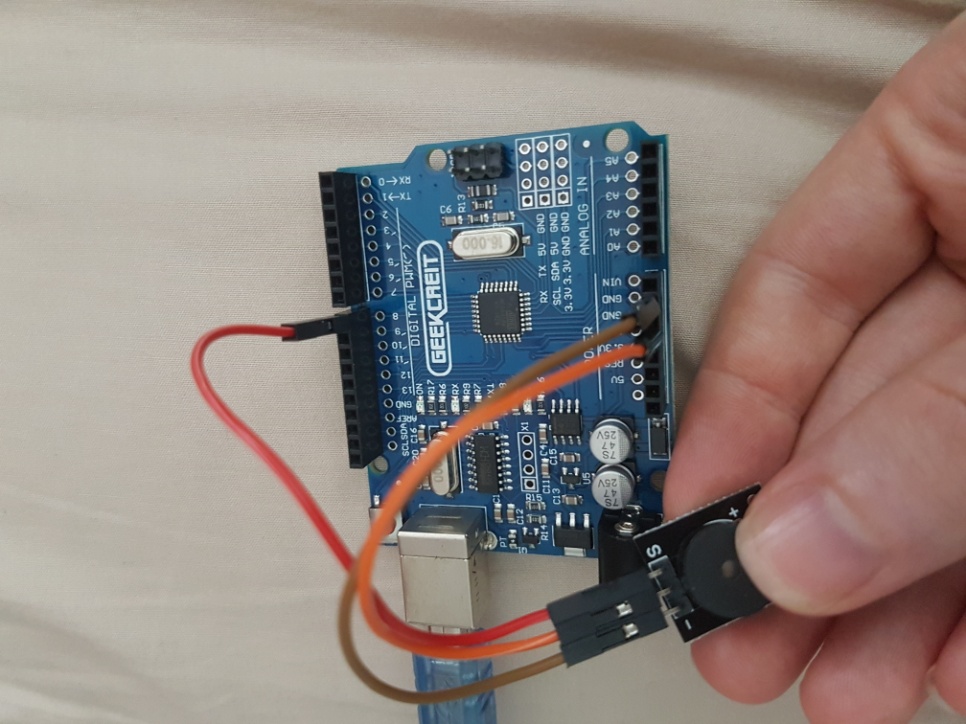
Steps:

1. Do the buzzer tutorial
2. Read the lab 2 specification
3. Create *playFrequency(double F, double M);* (see pseudo code)
4. Play a basic melody of your choice

­­­­Tutorial

This IC has 3 pins to attach to the Arduino, a signal pin which is connected to a digital output pin on the arduino (this is labelled ‘S’ on the buzzer), a VCC pin which connects to any of the 5 volt pins on the Arduino (this is the middle pin on the buzzer) and a GND pin which connects to any of the ground pins on the Arduino the pin (labelled ‘I’ or ‘-’ opposite the ‘S’ whichever way you want to look at it) .

A picture below shows my wiring, note the colours of the pin wires (called male to female wire connectors) do not mean anything. For this tutorial I am using digital pin 8 for the I/O pin (the ‘S’ pin).



The way the IC works is as follows, the I/O pin on the buzzer must receive a 1 followed by a 0 at some frequency f. The frequency defines the frequency of the sound the buzzer makes, eg. Middle-C (the middle key of C on the piano) has a frequency of 523.25.

int ioPin;

double frequency = 523.25; //middle-c

void setup() {

ioPin = 8;

pinMode(ioPin, OUTPUT);

}

void loop() {

double period = 1.0 / frequency;

double microseconds = 1000000.0 \* period;

microseconds \*= 0.5;

digitalWrite(ioPin, LOW);

delayMicroseconds((int)microseconds);

digitalWrite(ioPin, HIGH);

delayMicroseconds((int)microseconds);

}

In this program we setup 2 variables, ioPin specifies the pin we connected I/O to, and frequency is the frequency we change our input signal at. Note: as you can see we can initialize frequency/ioPin/any global variable in global space or in setup(). The inverse of the frequency is the period, this is the amount of time where the signal goes up then down. It is computed as 1.0 / frequency. Then in our loop function we compute the period as 1.0 / frequency, this is the amount of time we have to set the output low then raise it high. This figure is per seconds, however we need to set it based on microseconds otherwise the frequency will not be accurate enough. So the period is multiplied by the number of microseconds in 1 second (1000000) then we want the signal to be low half this time and high the rest so we multiply it by 0.5. Using the digitalWrite function we learnt, we set the pin LOW, then wait for the half period amount of time (in microseconds).

## Lab 2: Random Access Melodies

If you haven't run through the basics of the buzzer module, please read that section prior to beginning this lab. In this lab you are required to write a program to play a tune on your Arduino. Basic melodies are created by playing notes one after the other (each note is played for a specific amount of time and sometimes there is an intentional absence of sound for a specific amount of time. In music, there are several notes: A, B, C, D, E, F and G. These notes may also be flat or sharp. For example D-sharp lies between D and E. E-flat is the same as D-sharp, so sharp means a half step higher and flat means a half step lower. Moreover these notes may be played at different octaves (pitches) and each combination of note and octave corresponds to a single identifiable frequency.

Some notes do not have a sharp or a flat (only half a step separates B to C and E to F). Here is a full list of the 12 notes of each octave:

1. A
2. A-sharp/B-flat
3. B
4. C
5. C-sharp/D-flat
6. D
7. D-sharp/E-flat
8. E
9. F
10. F-sharp/G-flat
11. G
12. G-sharp/A-flat

This site contains a list of notes to frequencies for different octaves: <http://www.phy.mtu.edu/~suits/notefreqs.html>.

Here are some examples in the 5th octave:

|  |  |
| --- | --- |
| Note Name | Frequency |
| C | 523.25 |
| C-sharp / D-flat | 554.37 |
| D | 587.33 |
| D-sharp / E-flat | 622.25 |
| E | 659.25 |
| F | 698.46 |
| F-sharp / G-flat | 739.99 |
| G | 783.99 |
| G-sharp / A-flat | 830.61 |
| A | 880.00 |
| A-sharp / B-flat | 932.33 |
| B | 987.77 |

You may generate the frequency from the note based on the above table.

Alternatively, given the note name (A,B,C,D,E,F,G), whether the note is flat or sharp as well as the octave, the frequency may be computed. The method for computing this is can be found at this site: <http://www.phy.mtu.edu/~suits/NoteFreqCalcs.html>.

In this lab you will be required to write a function to play a note (or play nothing) for a specific amount of time. This function should have the signature:

voidplayNote(char note, chartype,int octave,intnumMilliseconds);

The function must play note 'note' at octave ‘octave’ with a type (‘ ’ (space) for no type, ‘s’ for sharp and ‘f’ for flat) for 'numMilliseconds' milliseconds. The note is input as a char type, this may be 'c', 'd', 'e', 'f', 'g', 'a', 'b' or the space character, ' ' (which plays nothing for the specified number of milliseconds). Variable type is also a char where a ‘ ’ (a space) signifies the note must be played as is, ‘f’ designates a flat note and ‘s’ designates a sharp note.

Using the playNote() function, play the opening part of jingle bells. We describe the melody using tuples of the form (note,millisecond) (note there are no flat or sharp notes here, try playing it at different octaves to view the results):

**Jingle-Bells : [(E,400), (E,400), (E,800), (E,400), (E,400), (E,800), (E,400), (G,400), (C,400), (D,400), (E,1600)]**

Next, you must choose a melody from a favourite song and research the notes and code the arduino to play it for you.

This is the pseudo code to play a sound at a particular frequency for M milliseconds (pseudo code looks like a programming language but isn’t and is simply meant as a guide for an algorithm for you to implement in the chosen language, for you this will be C):

//play frequency F, for M milliseconds

period = (1 / F) \* 1000000

durationMicroseconds = M \* 1000

time = 0

while time <durationMicroseconds

output HIGH voltage

delay for period / 2 microseconds

output LOW voltage

delay for period / 2 microseconds

time += period