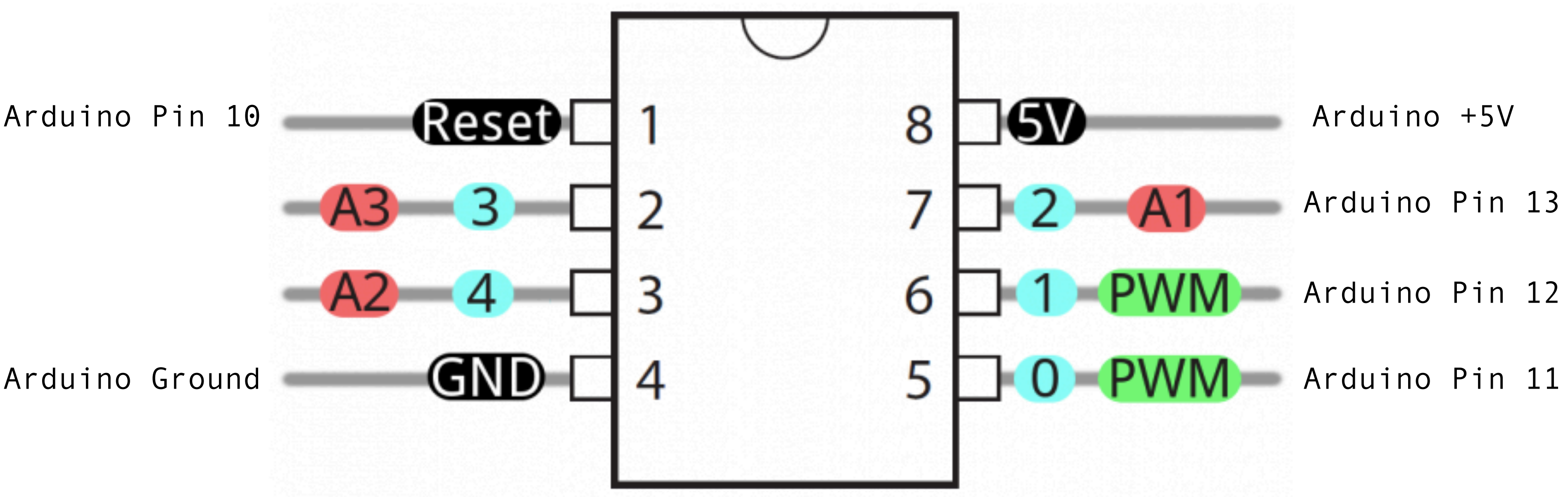
Electronics 2

Electronics for Fabricademy

Emma Pareschi

How to connect to Arduino Uno Programmer



Arduino Pin 10

Arduino Ground

Arduino +5V

Arduino Pin 13

Arduino Pin 12

Arduino Pin 11

DIGITAL PINS → digitalWrite / digitalRead

PWM PINS → analogWrite

ANALOG PINS → analogRead

Steps

Install the Attiny Board in Arduino IDE:
https://raw.githubusercontent.com/damellis/attiny/ide-1.6.x-boards-manager/package_damellis_attiny_index.json

http://drazzy.com/package_drazzy.com_index.json

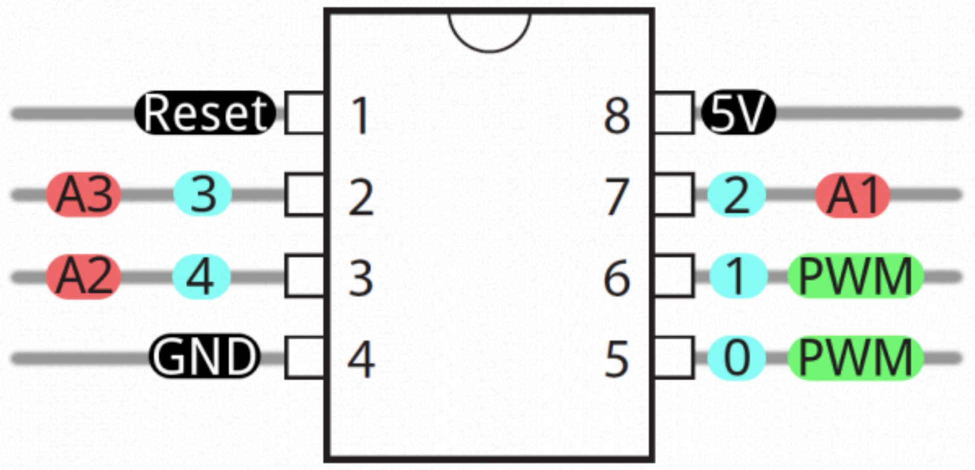
1) Program the Arduino to be a programmer
2) Connect the Attiny85
3) Select board (attiny85 (internal 8MHz)) and port
4) Select programmer: Arduino as ISP
5) Burn bootloader
6) Upload your code

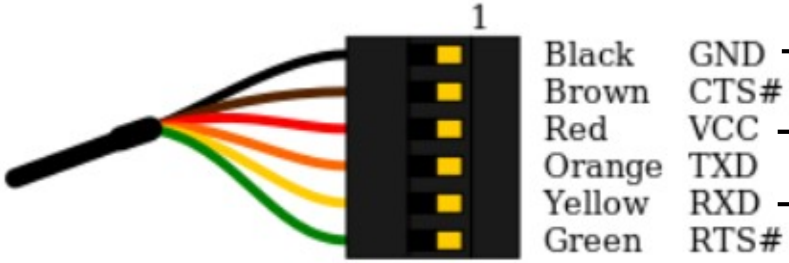
Resources:

- How install Attiny board in Arduino IDE:
  https://create.arduino.cc/projecthub/arjun/programming-attiny85-with-arduino-uno-afb829

- program Attiny with Arduino Uno:
  https://www.instructables.com/id/Program-an-ATtiny-with-Arduino/

How to print on Serial Monitor

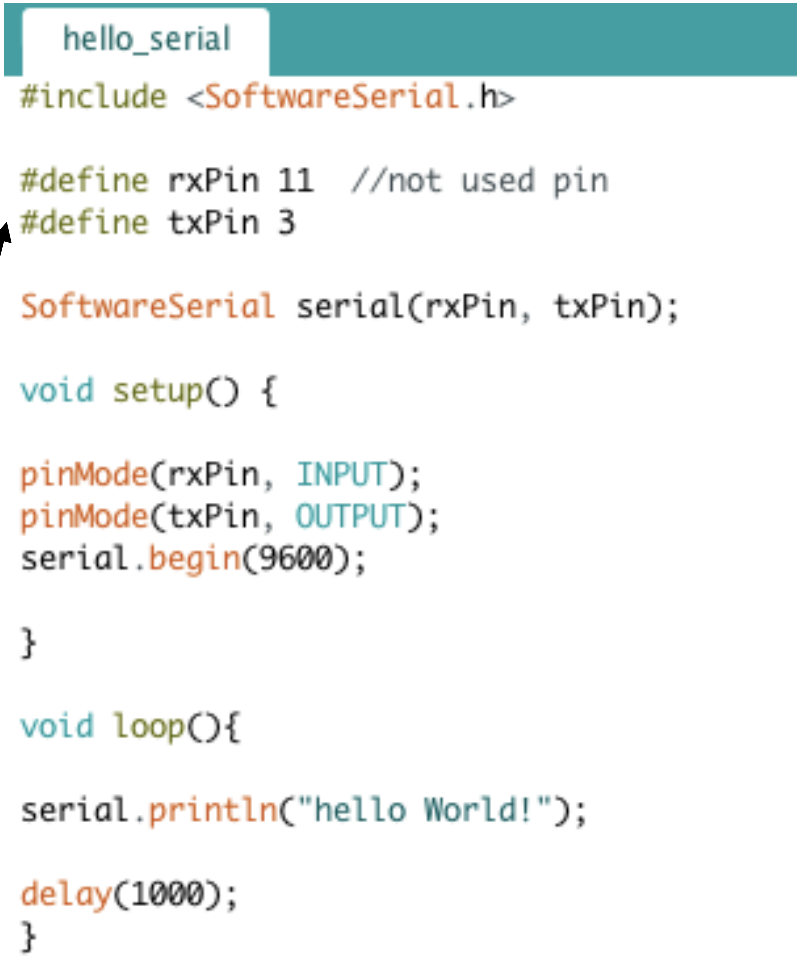




GND

5V

Pin 3

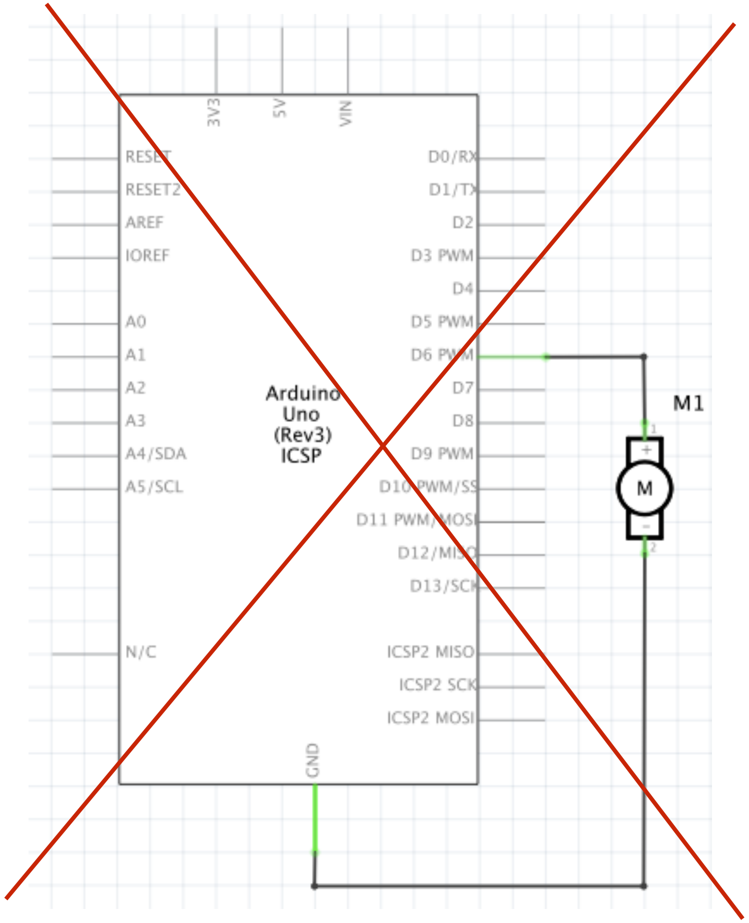


```
#include <SoftwareSerial.h>

#define rxPin 11  //not used pin
#define txPin 3

SoftwareSerial serial(rxPin, txPin);

void setup() {

pinMode(rxPin, INPUT);
pinMode(txPin, OUTPUT);
serial.begin(9600);

}

void loop(){

serial.println("hello World!");

delay(1000);
}
```

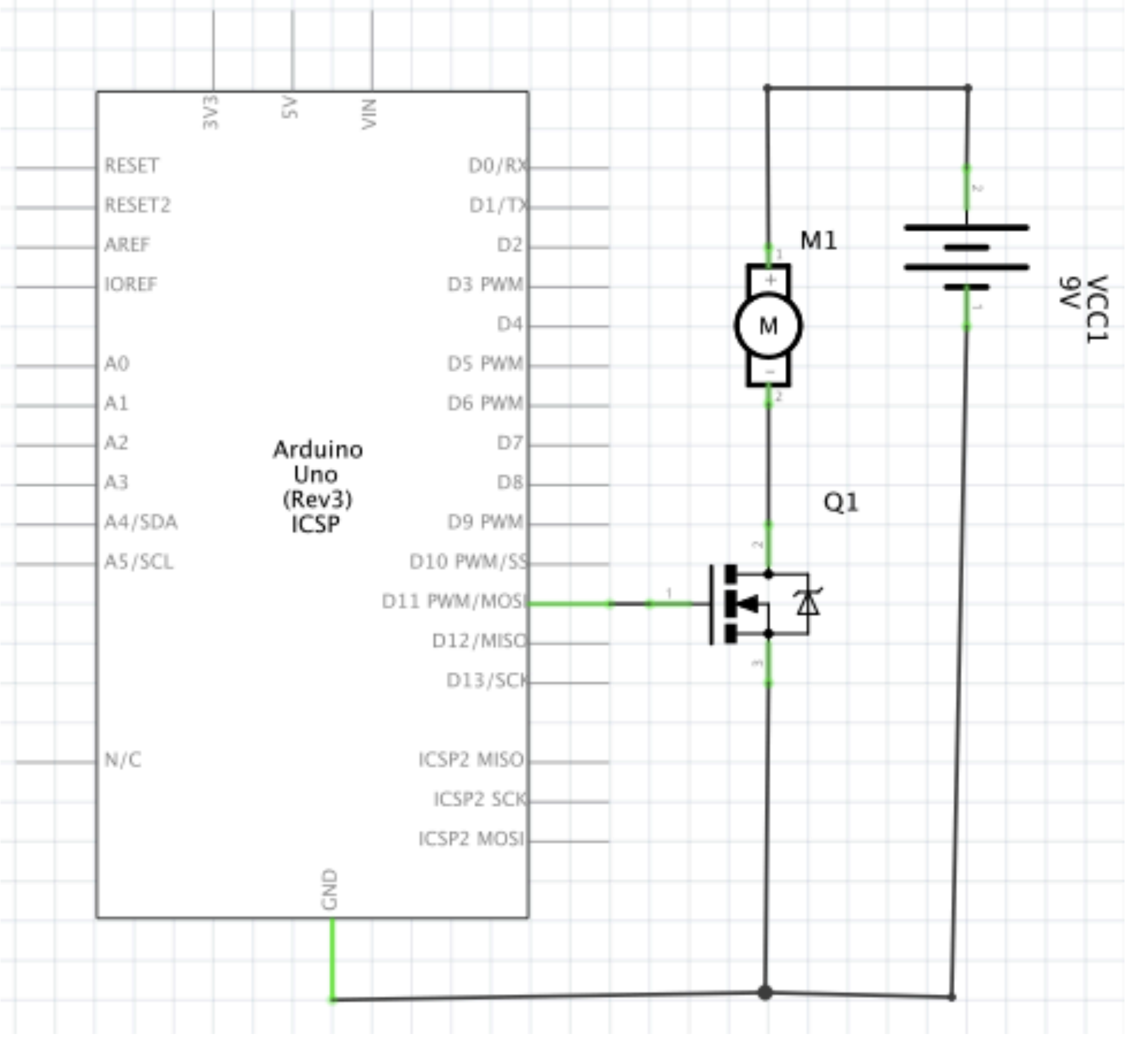
Power Load

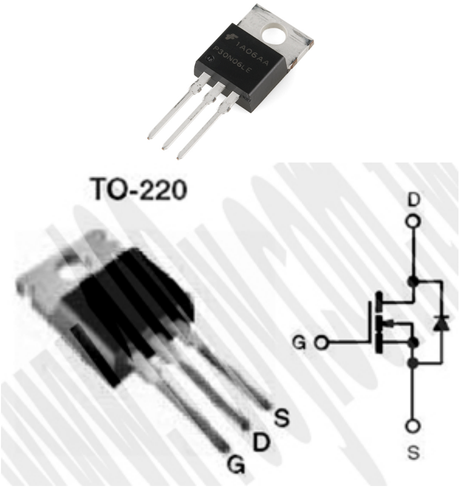
everything that sinks more than 40mA:

- Motors
- Conductive coils
- Heat element



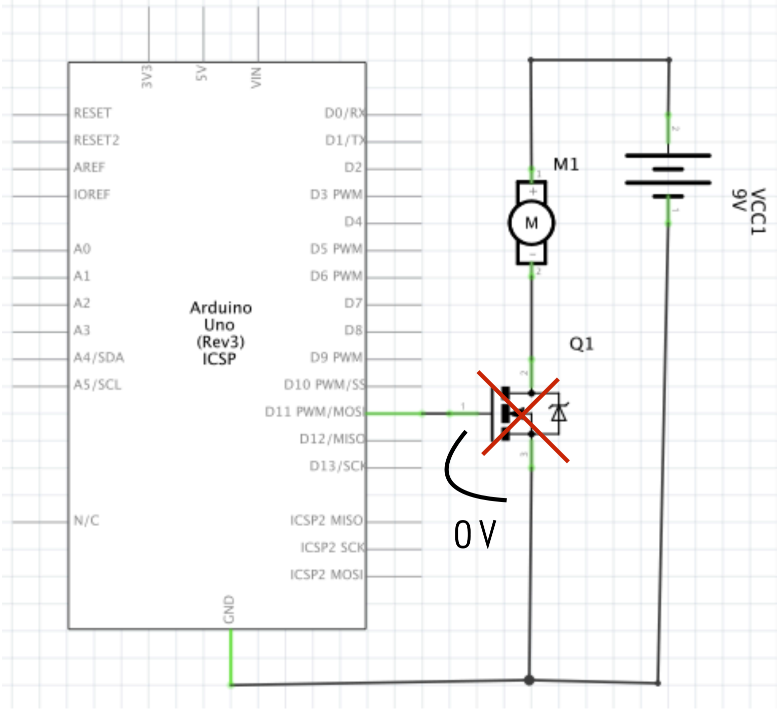
DC Motor (brushed) - Schematic



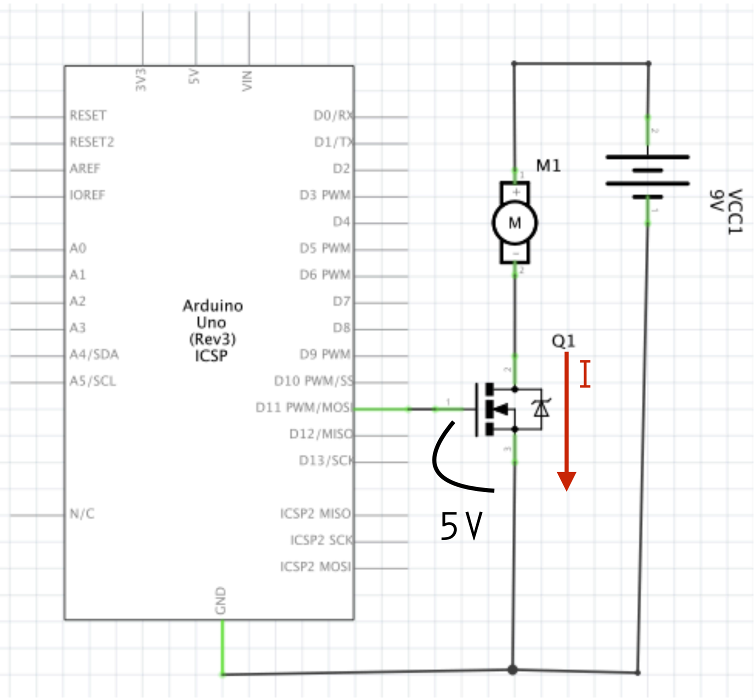


Three terminals:
Source
Gate
Drain

Mosfet N-channel



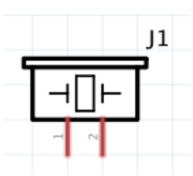
If Vgs = 0V
=> open loop, no current

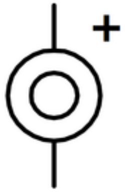


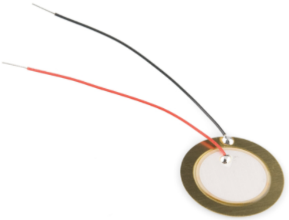
If Vgs = 5V
=> Close loop, current

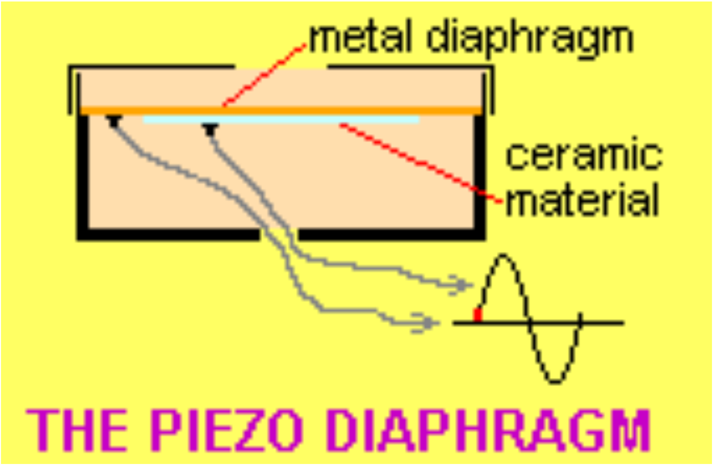
Transducer - PIEZOELECTRIC

A piezoelectric transducer contains a diaphragm consisting of a thin brass disc on which is mounted a ceramic wafer. When an AC signal is applied between the piezoelectric wafer and the disc, the disc flexes at that frequency.

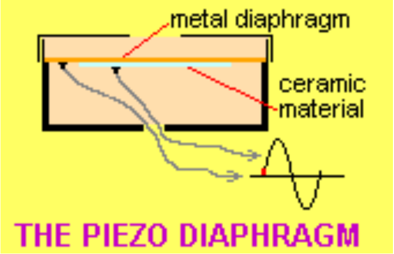








Frequency



Frequency is the number of occurrences of a repeating event per unit of time.[

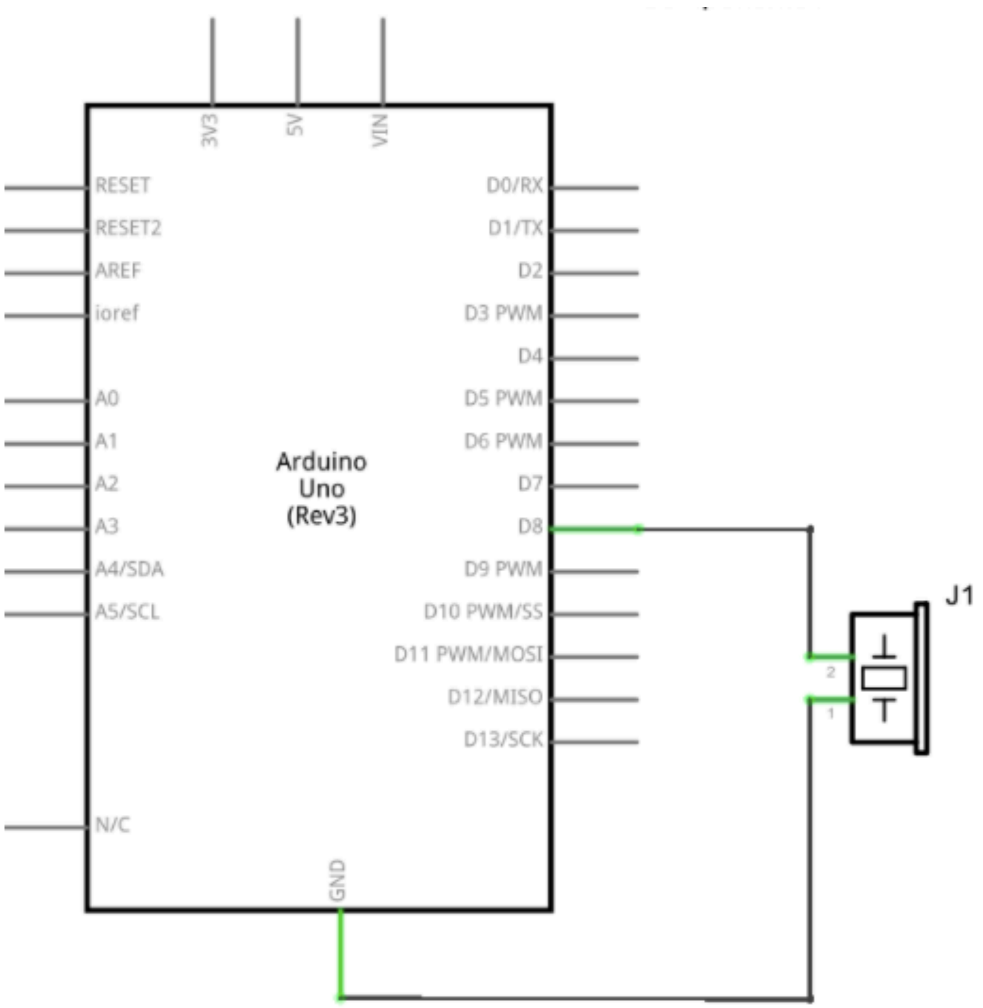
Audio frequency is measured in Hertz, abbreviated Hz, named after Heinrich Rudolf Hertz, the first scientist to prove the existence of electromagnetic waves.

The H in Hz is capitalised because it refers to a real name. One thousand Hertz can be written as 1 kiloHertz, almost always abbreviated as 1kHz (note that the k is lowercase).

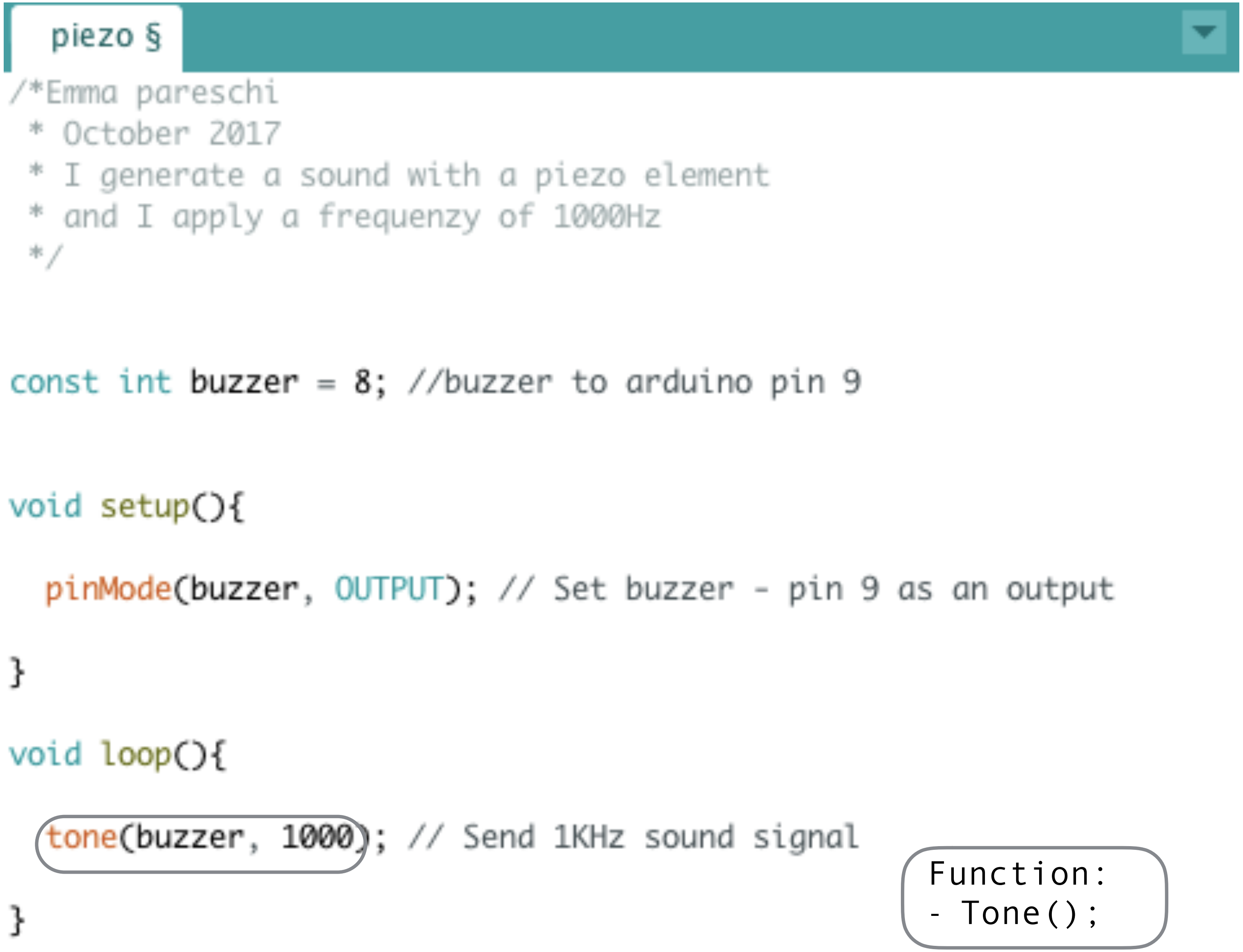
The human ear is often described as being able to detect sounds between 20Hz and 20kHz, although the ability to hear sounds above 15kHz is relatively unusual and diminishes naturally with age. Sensitivity to all frequencies can be impaired by long-term exposure to loud noise.

The most common frequencies applied to audio transducers range between 3kHz and 3.5kHz. Piezoelectric elements are inefficient for generating sounds below 1kHz.

Piezo schematic



The piezo is connected between pin 8 and ground



```
/*Emma pareschi
 * October 2017
 * I generate a sound with a piezo element
 * and I apply a frequenzy of 1000Hz
 */


const int buzzer = 8; //buzzer to arduino pin 9


void setup(){

  pinMode(buzzer, OUTPUT); // Set buzzer - pin 9 as an output

}

void loop(){

  tone(buzzer, 1000); // Send 1KHz sound signal

}
```

Function:
- Tone();

Function TONE

```
tone( pin number, frequency in hertz);
```

1. The pin number that you will use on the Arduino.
2. The frequency specified in hertz

```
tone( pin number, frequency in hertz, duration in milliseconds);
```

1. The pin number that you will use on the Arduino.
2. The frequency specified in hertz
3. The duration of the tone in milliseconds (optional) - unsigned long

```
const int buzzer = 8;

tone(buzzer, 1000);
tone(buzzer, 1000, 5000);
```

Notes:

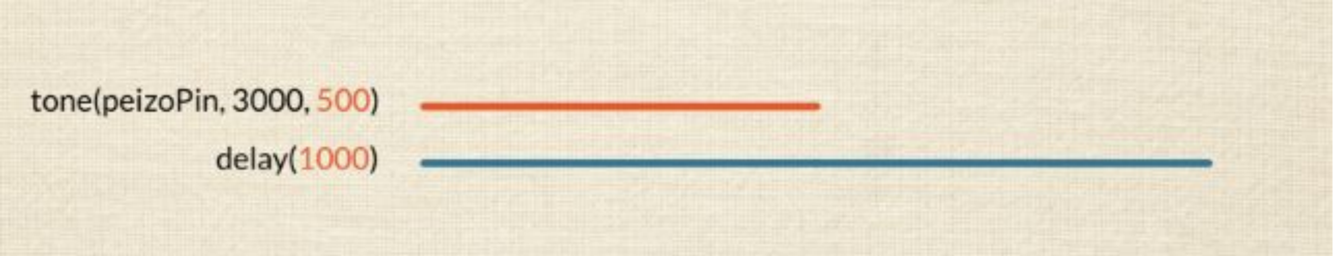
- min frequency: 31Hz
- Max frequency: 65535

Function TONE // noTone

```
const int buzzer = 8;

tone(buzzer, 1000, 1000);
delay(1000);
```

I want to generate distinct beats long 1sec
every 1sec.
I try to use delay.
But it doesn't work.
Why?

```
tone(buzzer, 3000, 500);
delay(1000);
```



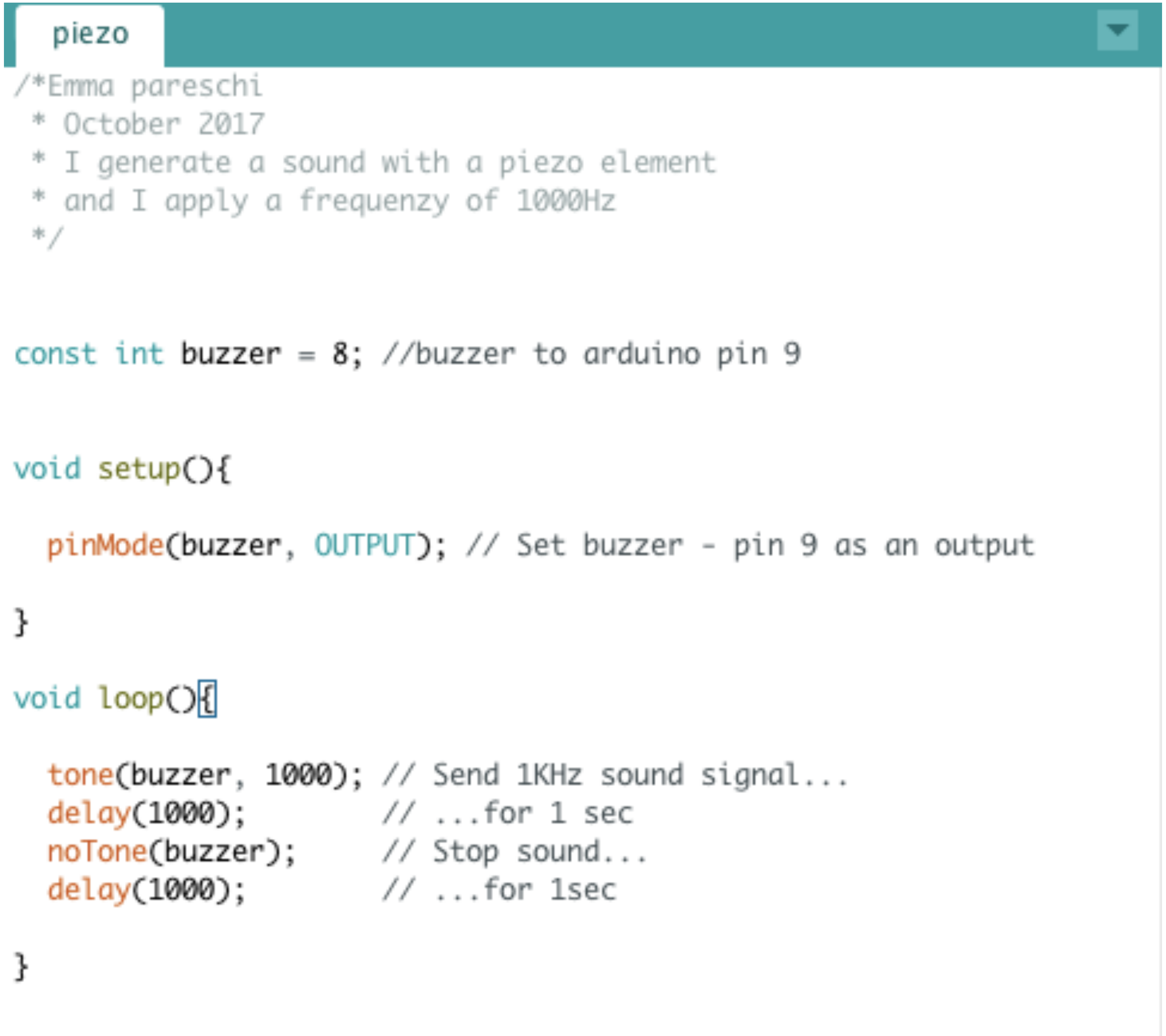
```
tone(peizoPin, 3000, 500)
delay(1000)
```

```
tone(buzzer, 1000); // Send 1KHz sound signal...
delay(1000);        // ...for 1 sec
noTone(buzzer);     // Stop sound...
delay(1000);        // ...for 1sec
```

```
noTone( pin number);
```

Stops the generation
of a square wave
triggered by `tone()`

Piezo sketch



```
/*Emma pareschi
 * October 2017
 * I generate a sound with a piezo element
 * and I apply a frequenzy of 1000Hz
 */


const int buzzer = 8; //buzzer to arduino pin 9


void setup(){

  pinMode(buzzer, OUTPUT); // Set buzzer - pin 9 as an output

}

void loop(){

  tone(buzzer, 1000); // Send 1KHz sound signal...
  delay(1000);        // ...for 1 sec
  noTone(buzzer);     // Stop sound...
  delay(1000);        // ...for 1sec

}
```

Limits of the Tone Function

1. You can't use tone() while also using analogWrite() on pins 3 or 11. If you do - you get some whacky results - neither will work like you expect. That's because the tone() function uses the same built in timer that analogWrite() does for pins 3 and 11. It's worth trying just hear the weird noises.

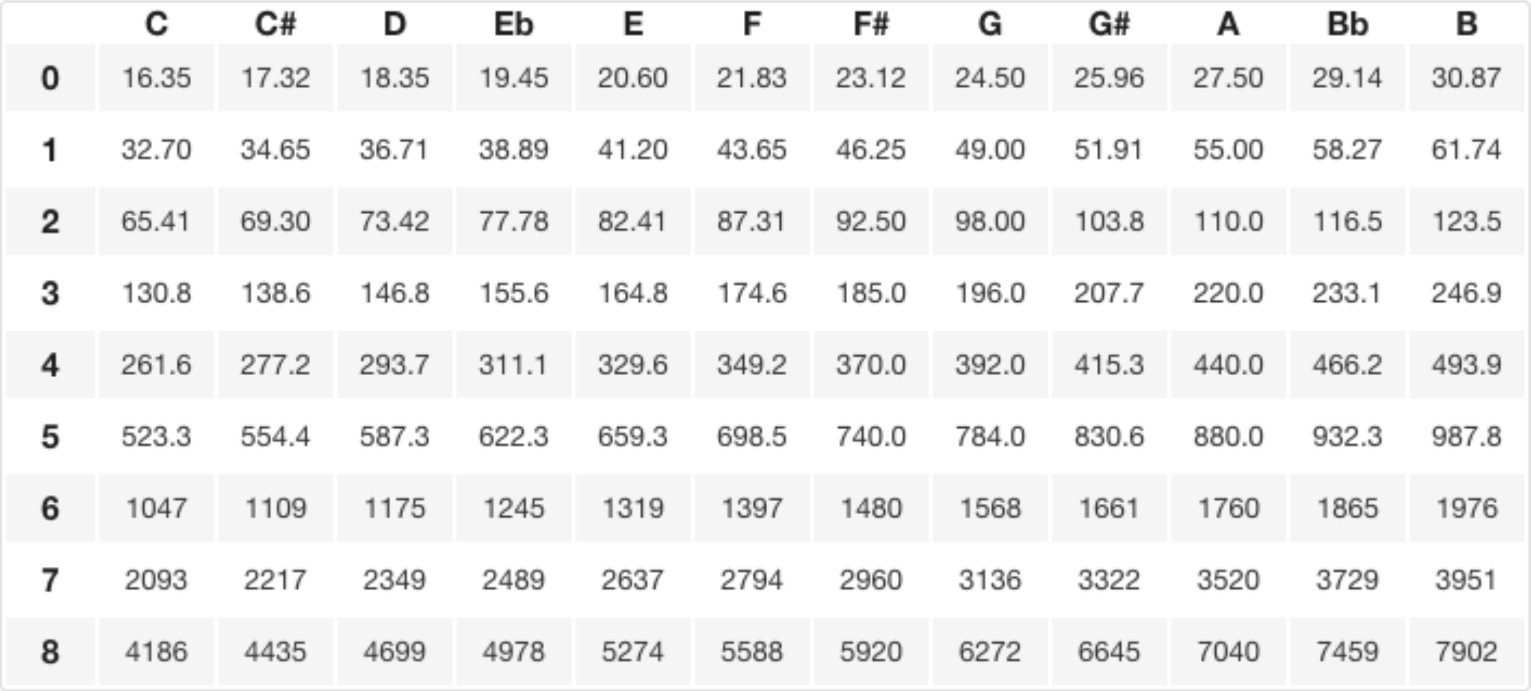
2. You cannot generate a tone lower than 31 HZ. You can pass values 31 and less to the tone() function, but it doesn't mean you will get a good representation of it.

3. The tone() function cannot be used by two separate pins at the same time. Let's say you have two separate piezo speakers, each on a different pin. You can't have them both play at the same time. One has to be on, and then the other. Furthermore, before you can have the other pin use the tone() function, you must call the noTone() function and "turn off" the tone from the previous pin.

Melody

To create a melody you need notes. Each note is defined by a frequency.

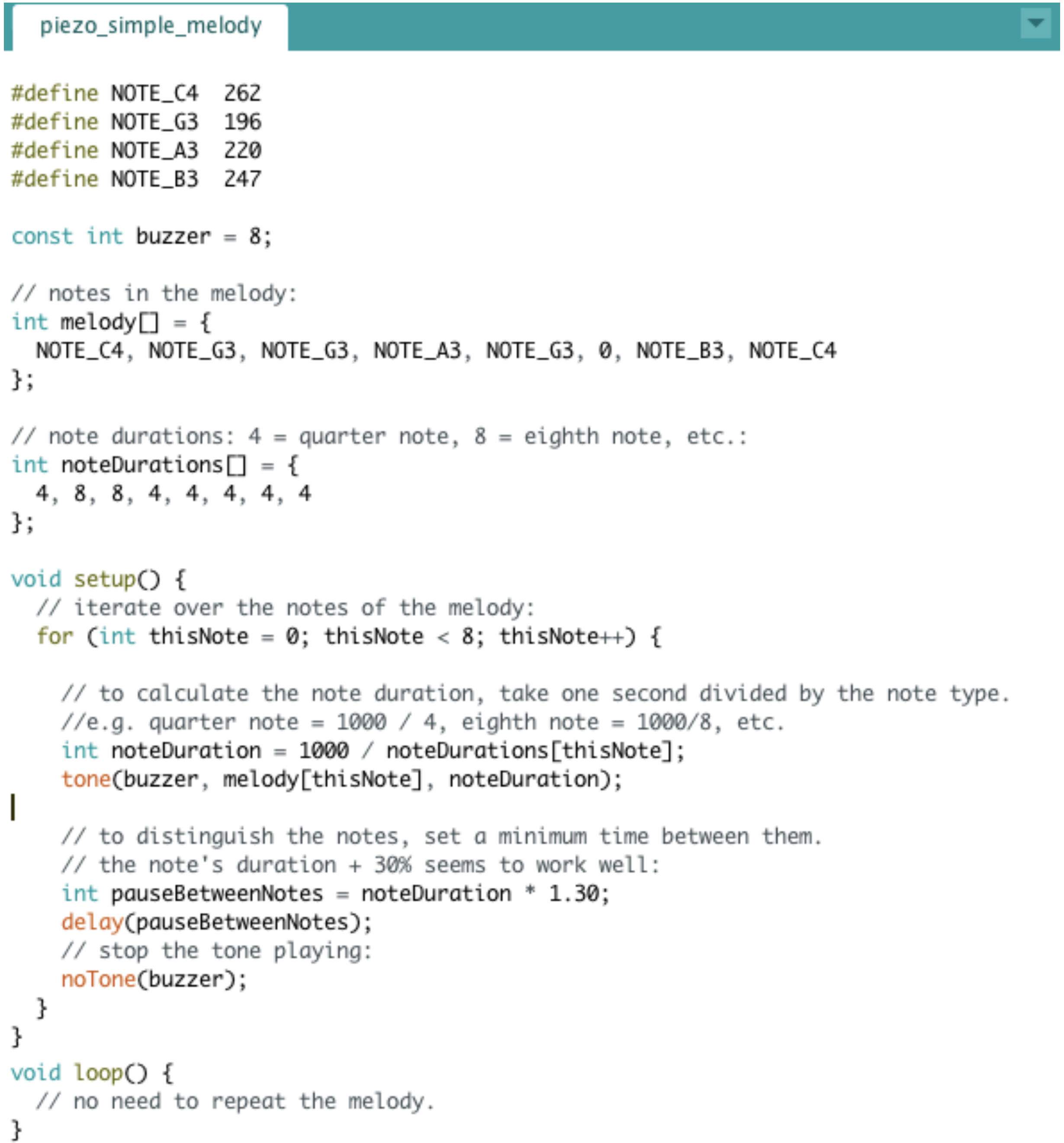
Note Frequency



	C	C#	D	Eb	E	F	F#	G	G#	A	Bb	B
0	16.35	17.32	18.35	19.45	20.60	21.83	23.12	24.50	25.96	27.50	29.14	30.87
1	32.70	34.65	36.71	38.89	41.20	43.65	46.25	49.00	51.91	55.00	58.27	61.74
2	65.41	69.30	73.42	77.78	82.41	87.31	92.50	98.00	103.8	110.0	116.5	123.5
3	130.8	138.6	146.8	155.6	164.8	174.6	185.0	196.0	207.7	220.0	233.1	246.9
4	261.6	277.2	293.7	311.1	329.6	349.2	370.0	392.0	415.3	440.0	466.2	493.9
5	523.3	554.4	587.3	622.3	659.3	698.5	740.0	784.0	830.6	880.0	932.3	987.8
6	1047	1109	1175	1245	1319	1397	1480	1568	1661	1760	1865	1976
7	2093	2217	2349	2489	2637	2794	2960	3136	3322	3520	3729	3951
8	4186	4435	4699	4978	5274	5588	5920	6272	6645	7040	7459	7902

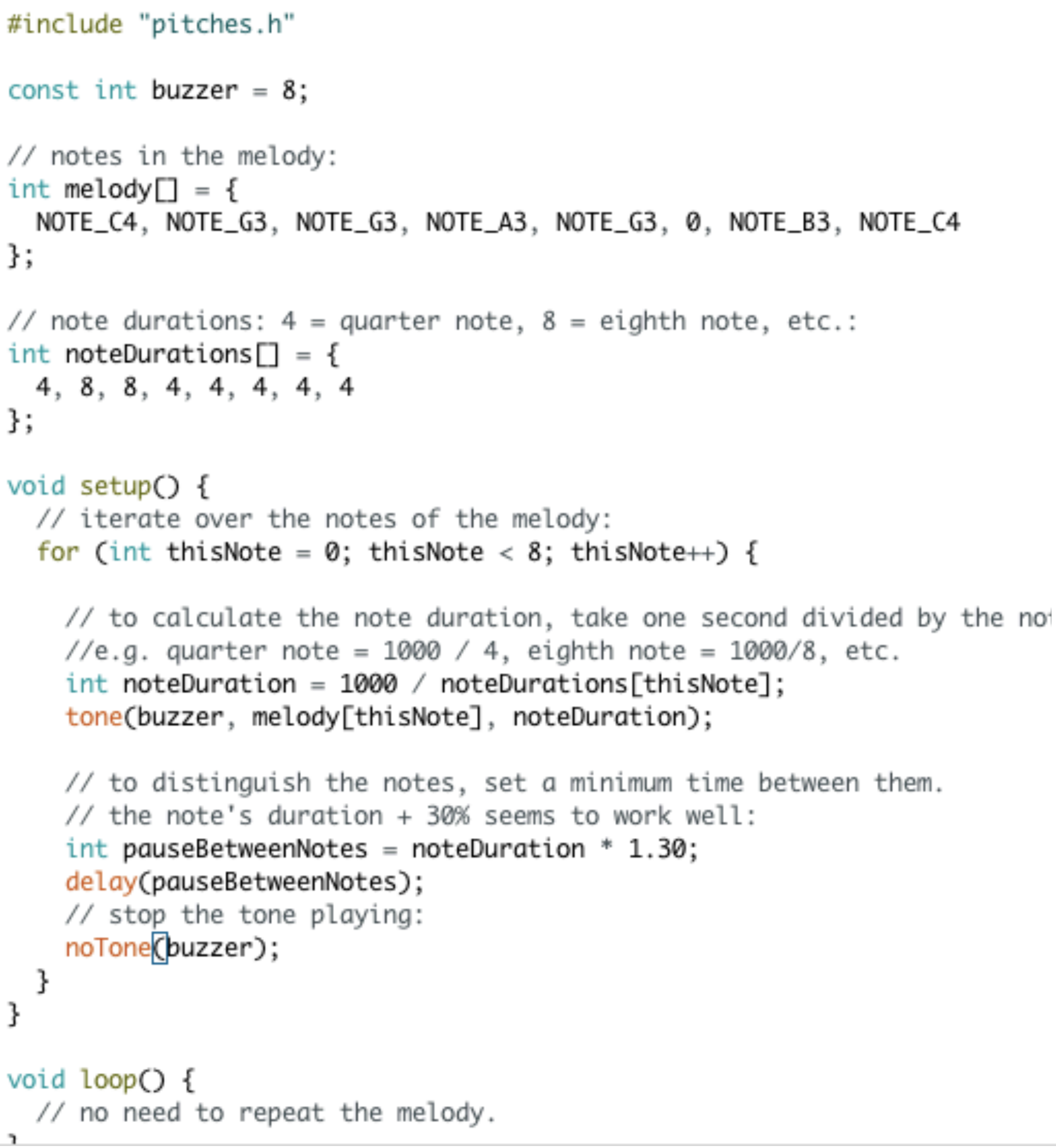
```
#define NOTE_C4  262
#define NOTE_G3  196
#define NOTE_A3  220
#define NOTE_B3  247
```

Melody

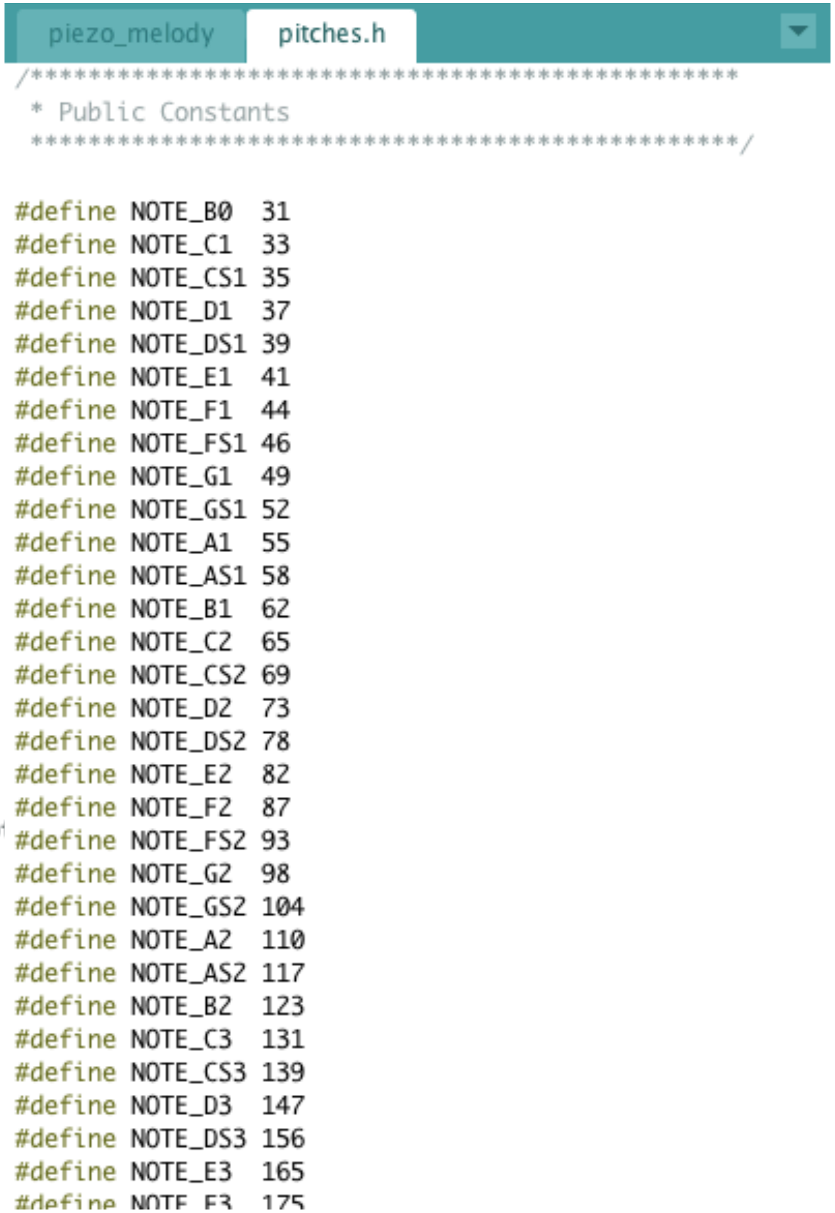


```
#define NOTE_C4  262
#define NOTE_G3  196
#define NOTE_A3  220
#define NOTE_B3  247

const int buzzer = 8;

// notes in the melody:
int melody[] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
  4, 8, 8, 4, 4, 4, 4, 4
};

void setup() {
  // iterate over the notes of the melody:
  for (int thisNote = 0; thisNote < 8; thisNote++) {

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

    // to distinguish the notes, set a minimum time between them.
    // the note's duration + 30% seems to work well:
    int pauseBetweenNotes = noteDuration * 1.30;
    delay(pauseBetweenNotes);
    // stop the tone playing:
    noTone(buzzer);
  }
}
void loop() {
  // no need to repeat the melody.
}
```

Melody



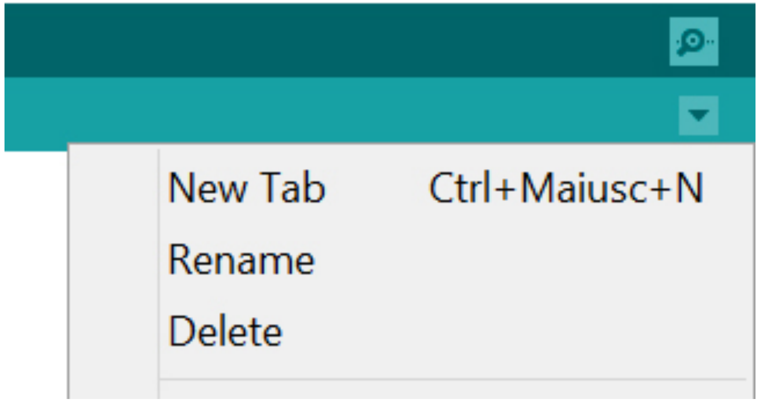
```
#include "pitches.h"

const int buzzer = 8;

// notes in the melody:
int melody[] = {
  NOTE_C4, NOTE_G3, NOTE_G3, NOTE_A3, NOTE_G3, 0, NOTE_B3, NOTE_C4
};

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
  4, 8, 8, 4, 4, 4, 4, 4
};

void setup() {
  // iterate over the notes of the melody:
  for (int thisNote = 0; thisNote < 8; thisNote++) {

    // to calculate the note duration, take one second divided by the not
    //e.g. quarter note = 1000 / 4, eighth note = 1000/8, etc.
    int noteDuration = 1000 / noteDurations[thisNote];
    tone(buzzer, melody[thisNote], noteDuration);

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

void loop() {
  // no need to repeat the melody.
}
```

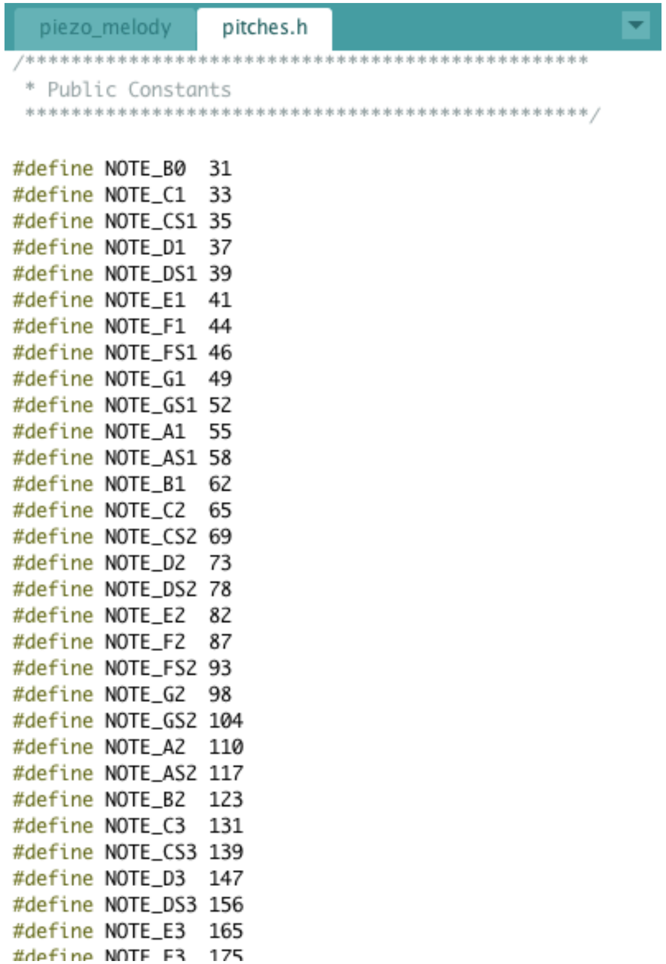


```
/*************************************************
 * Public Constants
 *************************************************/

#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
```

Melody

To create the file pitches,
select new tab:





```
/*************************************************
 * Public Constants
 *************************************************/

#define NOTE_B0  31
#define NOTE_C1  33
#define NOTE_CS1 35
#define NOTE_D1  37
#define NOTE_DS1 39
#define NOTE_E1  41
#define NOTE_F1  44
#define NOTE_FS1 46
#define NOTE_G1  49
#define NOTE_GS1 52
#define NOTE_A1  55
#define NOTE_AS1 58
#define NOTE_B1  62
#define NOTE_C2  65
#define NOTE_CS2 69
#define NOTE_D2  73
#define NOTE_DS2 78
#define NOTE_E2  82
#define NOTE_F2  87
#define NOTE_FS2 93
#define NOTE_G2  98
#define NOTE_GS2 104
#define NOTE_A2  110
#define NOTE_AS2 117
#define NOTE_B2  123
#define NOTE_C3  131
#define NOTE_CS3 139
#define NOTE_D3  147
#define NOTE_DS3 156
#define NOTE_E3  165
#define NOTE_F3  175
```