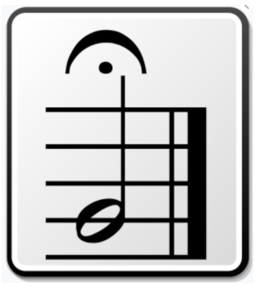
Systems and Industrial Engineering
University of Arizona

SIE 370 Lab

Generating Sound

1. Objectives

In this lab, you will create a prototype for an embedded system controller that plays music using pre-defined pitches and another that controls and issues an alarm. Both of these experiments will be prototyped first using Tinkercad.



2. Lab Reading Assignment

2.1 Reading Assignment:
Textbook: Exploring Arduino, Jeremy Blum, Chapter 6 Making Sounds and Music

Topics include:

- Properties of Sound
- How a Speaker Produces Sound
- Arduino function tone()
  - https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/
- Including a definition (.h) file
- Using Arrays
- Debouncing a button
  - From our previous reading in Blum's Chapter 2 Exploring Arduino

2.2 Guiding Questions

Use these guiding questions are done in preparation for the lab. If you discover something during your reading or prototyping you may want to write about that in your Lab Report.

(a) Using our Arduino Reference: https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/ , we see that there are 3 parameter values for the Arduino *tone()* function. What are they and what do they represent?
(b) In our textbook reading, Blum mentions 3 limitations of the Arduino *tone()* function. What are they?

(c) We are using a modified version of Blum's *Boolean function debounce()*. Why do we need this?

(d) Blum explains Pull-up and Pull Down Resistors. What purpose do they serve and which one are we using in this Lab and why?

(e) Switch Bouncing
   1. Explain what it is,
   2. Explain what you did in Task 2 to avoid it, and
   3. Why it worked.

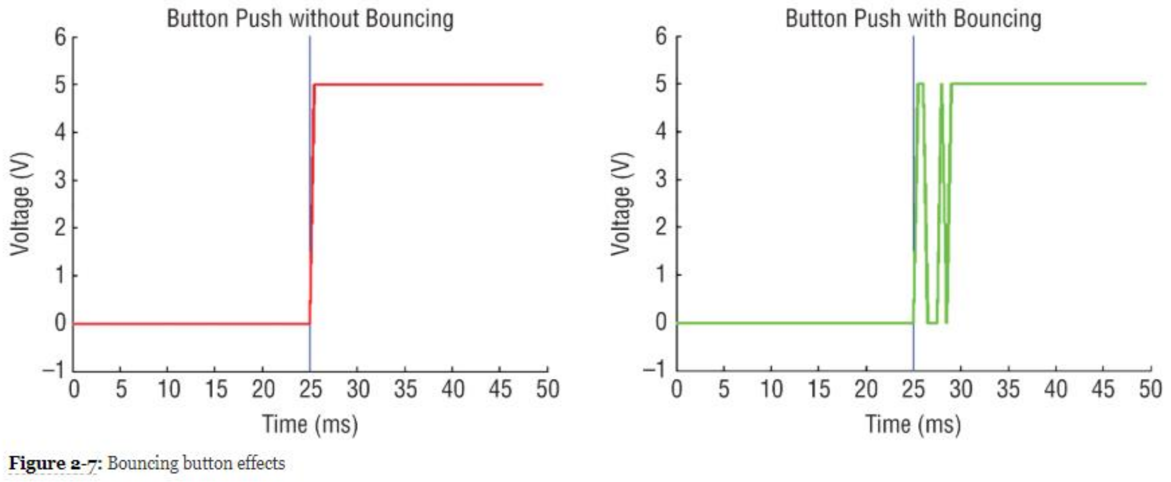


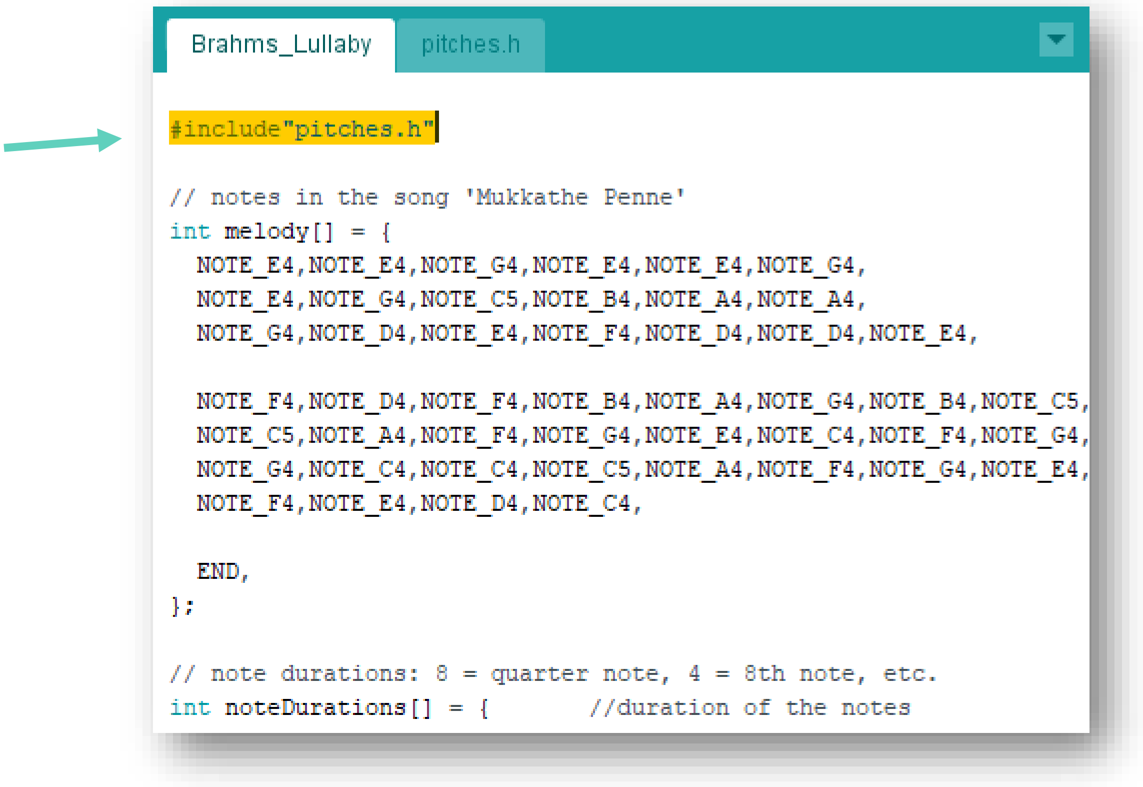
Figure 1 Switch bouncing effects

3. Task Prototyping

3.1 Task 1: Brahms' Lullaby Circuit

Re-create the In-Lab Task 3 Brahms' Lullaby Circuit and software using the instructions below, but adapt the instructions for Tinkercad. Test your wiring and code here in preparation for Lab and to answer any questions you may be asked. Since you were given code for this task, please make sure you understand what each line does before you come to lab.

For this software to work, there is a small change you will have to make *only* in Tinkercad. Tinkercad does not allow header files, so you will have to copy the contents of Pitches.h into the main program at the top of the file. Make sure you copy it into and replace the line (directive) for including the header file as shown below. This is exactly what the compiler will do with your **#include directive**.



```
#include"pitches.h"

// notes in the song 'Mukkathe Penne'
int melody[] = {
  NOTE_E4,NOTE_E4,NOTE_G4,NOTE_E4,NOTE_E4,NOTE_G4,
  NOTE_E4,NOTE_G4,NOTE_C5,NOTE_B4,NOTE_A4,NOTE_A4,
  NOTE_G4,NOTE_D4,NOTE_E4,NOTE_F4,NOTE_D4,NOTE_D4,NOTE_E4,

  NOTE_F4,NOTE_D4,NOTE_F4,NOTE_B4,NOTE_A4,NOTE_G4,NOTE_B4,NOTE_C5,
  NOTE_C5,NOTE_A4,NOTE_F4,NOTE_G4,NOTE_E4,NOTE_C4,NOTE_F4,NOTE_G4,
  NOTE_G4,NOTE_C4,NOTE_C4,NOTE_C5,NOTE_A4,NOTE_F4,NOTE_G4,NOTE_E4,
  NOTE_F4,NOTE_E4,NOTE_D4,NOTE_C4,

  END,
};

// note durations: 8 = quarter note, 4 = 8th note, etc.
int noteDurations[] = {       //duration of the notes
```

Figure 2 Brahms_Lullaby Program Showing the Include Command for Pitches.h

3.2 Task 2 Piezo Alarm Circuit

Re-create the In-Lab Task 4 Piezo Alarm Circuit and software using the instructions for the lab as written below but adapt the instructions for Tinkercad. Test your wiring and code here in preparation for Lab and to answer any questions you may be asked.

4. Instructions for Lab Task Experiment

In this experiment, you will need to perform several tasks. Please do them in order. After each successful completion of a task, have the instructor check your work so you can get credit and move on to the next task.

4.1 Component List

In addition to your laptop, Arduino, and a breadboard you will need the components in the list below.

Component	Quantity
Piezo Buzzer	1
Red LED	1
Push Button Switch	1
1 kΩ Resistor	1
100 - 220 Ω Resistor	1

Figure 3 Lab Component List

4.2 Task 3: Brahms' Lullaby Circuit

In your Tinkercad version, you should have already tested your Brahms' Lullaby Player program. If it worked, this task should just be creating the circuit for a Piezo buzzer (see image below) and testing both the circuit and the code. However, this time you will need to use the pitches.h file and make sure you place the **#include directive** in your Arduino sketch.

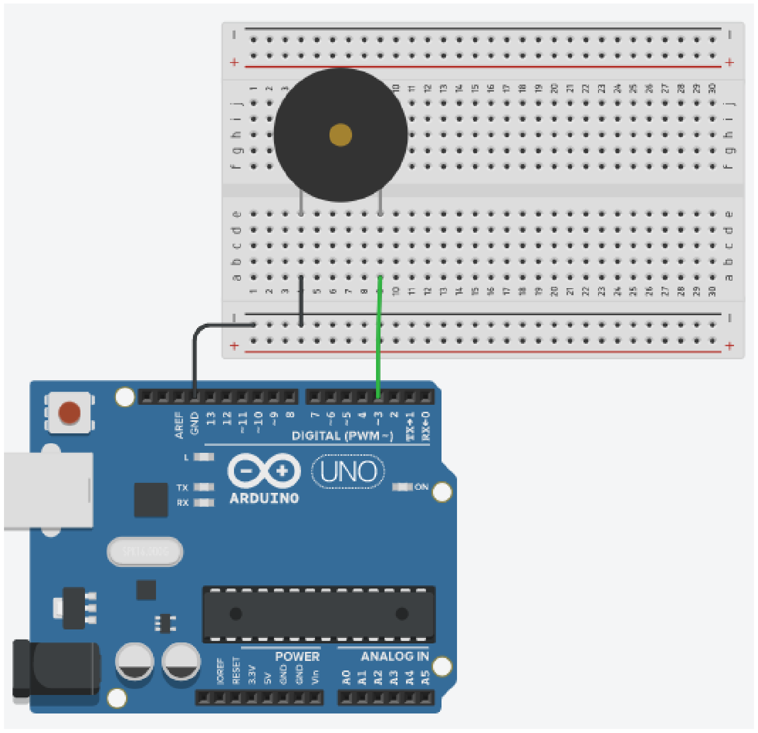


Figure 4 Piezo Buzzer for Brahms' Lullaby Circuit

Connections for this task are quite simple, as shown in the image above:

- The Breadboard Ground Rail is connected to the Arduino GND,
- The positive lead on the buzzer is connected to Arduino Pin 3
- The negative lead on the buzzer is connected to the Ground Rail.

The software we will use to control the operation of this circuit can be found in Brahms_Lullaby.ino and pitches.h. You will need to make sure the .ino program is using the **#include directive** to load the contents of the file. Run and test this file and change the value for the speed variable a few times to see if your program output is affected in the same way that Blum describes in Chapter 6.

When you have completed this task, show your Instructor your working project.

Since you were given code for this task, you do not have to hand it in, but please make sure you understand what each line does before you come to lab.

4.3 **Task 4: Piezo Alarm Circuit**

In your prototyping task, you should have already tested your Piezo Alarm program.

Build a circuit on your protoboard using a push button (tact switch), a Piezo Buzzer and an LED. In operation, you will use the switch to turn the alarm on and turn the alarm off.

When the program is run, the default position for the alarm will be off. When the button is pushed, the alarm sound will be activated, and the red LED will be lit. When the button is pushed again, the alarm sound will be turned off and the red LED will go dark.

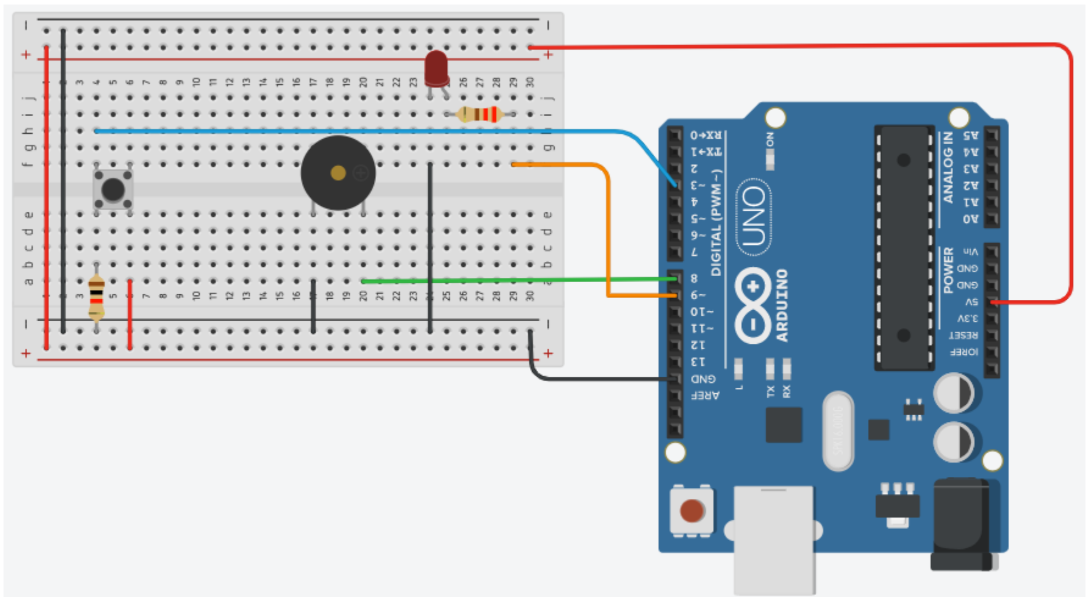


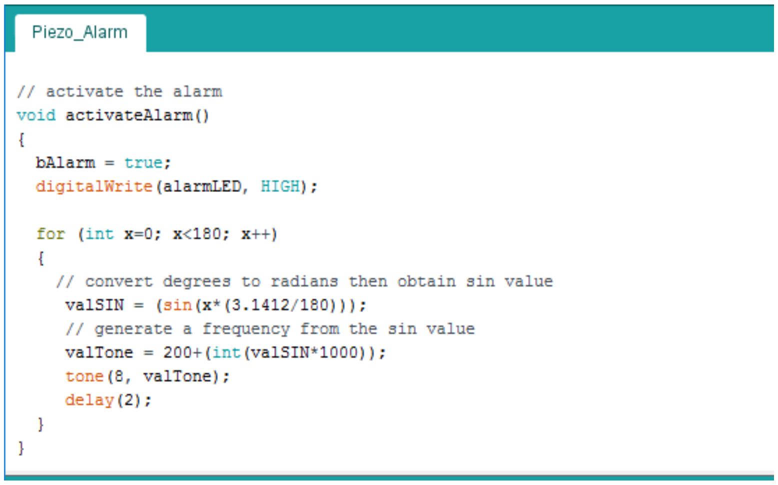
Figure 5 Piezo Alarm Circuit

Connections for this task as shown in the image above:

- The Breadboard Ground Rail is connected to the Arduino GND,
- The positive lead on the buzzer is connected to Arduino Pin 8,
- The negative lead on the buzzer is connected to the Ground Rail,
- The lower left terminal on the Push Button switch is connected to a Pull-down resistor (1 kΩ Resistor),
- The lower right terminal on the Push Button switch is connected to the Breadboard Power Rail,
- The upper left terminal on the Push Button switch is connected to Pin 3,
- The cathode of the red LED is connected to the Breadboard Ground Rail, and
- The anode of the red LED is connected to a 100 - 220 Ω Resistor (or a resistor in the 100 - 220 Ω range) and then Pin 9.

The software we will use to control the operation of this circuit can be found in Piezo_Alarm.txt. You should copy this program into your Arduino IDE and save it as Piezo_Alarm.ino for uploading to the Arduino. This will create a folder for the program, which is fine.

Inside the program there is a function called activateAlarm(), shown below



```
// activate the alarm
void activateAlarm()
{
  bAlarm = true;
  digitalWrite(alarmLED, HIGH);

  for (int x=0; x<180; x++)
  {
    // convert degrees to radians then obtain sin value
     valSIN = (sin(x*(3.1412/180)));
     // generate a frequency from the sin value
     valTone = 200+(int(valSIN*1000));
     tone(8, valTone);
     delay(2);
  }
}
```

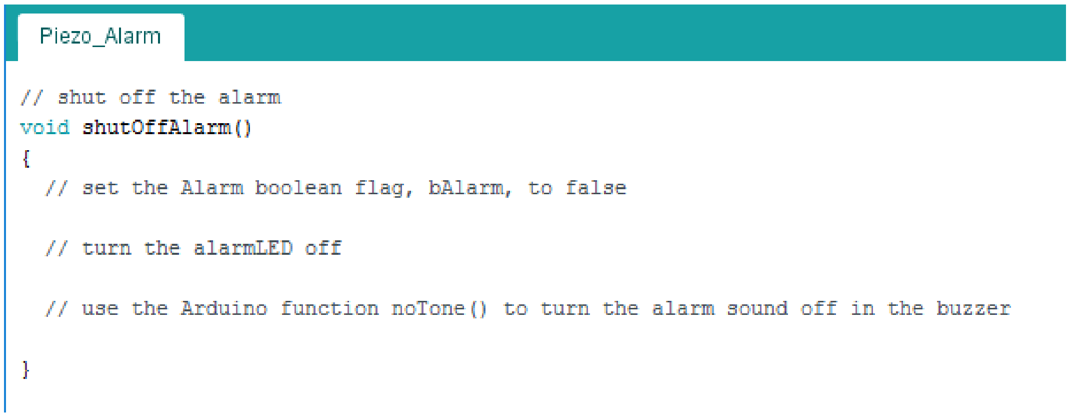
Figure 6 Piezo_Alarm.ino showing the activateAlarm() function

The concept here is that we are creating a sine wave and having the sound from the Piezo buzzer follow the path of that wave.

We use the sin() function, which is a mathematical function for the sine of an angle. We need to give the function the angle, expressed in radians. We store that radian value into our valSIN variable. We have a for loop that goes from 0 to 179; we don't want to go past halfway as this will take us into negative values. The sound will rise and fall, similar to a car alarm.

The valTone float variable takes the value in valSIN, and converts it to a frequency we require. We begin with 2,000 and add the valSIN multiplied by 1,000 giving us a good range of frequencies for the rising and falling tone. Digital pin 8 will be our output.

Write the code for the Alarm Shut Off function, shutOffAlarm(), shown below that will:
a) Turn off the Alarm flag
b) Turn off the Alarm LED
c) Use the noTone() function to stop the alarm sound



```
// shut off the alarm
void shutOffAlarm()
{
  // set the Alarm boolean flag, bAlarm, to false

  // turn the alarmLED off

  // use the Arduino function noTone() to turn the alarm sound off in the buzzer

}
```

Figure 7 Piezo_Alarm.ino showing the shutOffAlarm() function

When you have completed this task, show your Lab Instructor your working project.

Since you were responsible for writing the code for the shutOffAlarm() function, you need to hand it in. You were given most of the code, however, so please make sure you understand what each line does before you come to lab.