Arduino SoundBoard Documentation

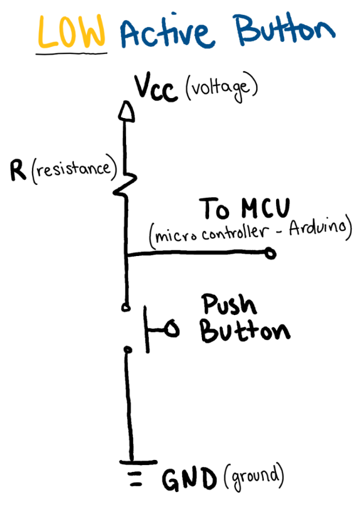
This is a step by step guide on starting your very own Arduino sound Board! This project will teach you how to use buttons and produce different sounds of your choosing using a speaker!

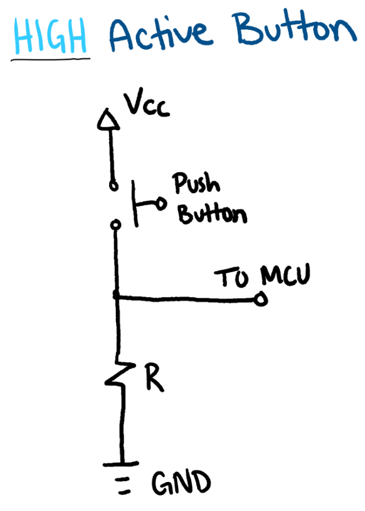
Here is some background information before starting this project:

Background →

Buttons:

There are two options for buttons that we could have used, a low active button or a high active button and the schematics are shown below. This soundboard circuit relies on the use of many buttons and for this design we decided to use **a low active button** to make circuit connections and coding a bit easier.

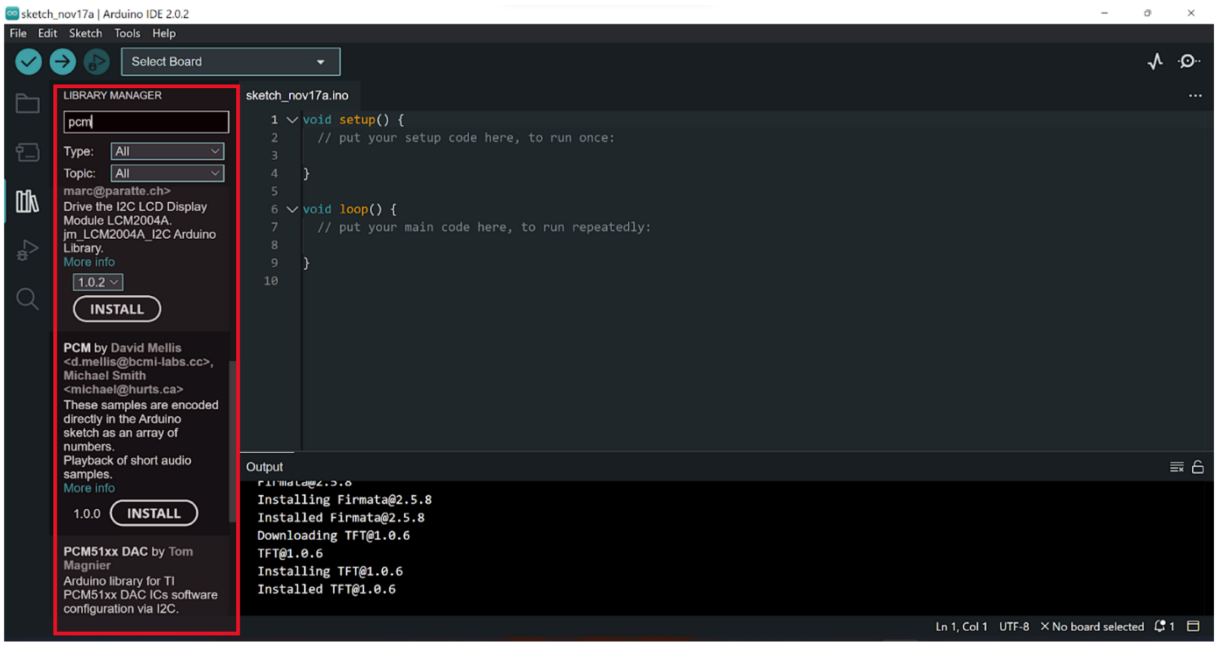




When the microcontroller (Arduino) receives a certain voltage it reads it as a high signal (or a 1), otherwise it receives a low signal or a 0.These zeros and ones can be programmed to mean whatever the programmer would like. In the cases of using buttons, a high active button works intuitively, outputting a high once the button is pressed and outputting a low when it isn't. When the button is not pressed there is an open circuit from the source (Vss) to the microcontroller (MCU),and the current is forced to flow through to ground, this outputs a 0 to the MCU. When it is pressed however the current flows through the branch of least resistance which is through the button, sending a voltage to the MCU indicating a 1. The low active button works exactly in the opposite way.

When designing our button we decided on the use of a low active button, this way it was easier to implement and code. In the code we forced the button to output a high at all times until a ground is introduced. So when constructing the button we used one wire to connect the MCU to the button and the other wire from the other end of the button to ground. This way the button is on HIGH at all times until the button is pressed then it connects straight to ground indicating a LOW or a zero.

Libraries :
Libraries are used in arduino to provide different tools that aren't built in functions that are always given. In this arduino project we need to use different sounds for each button so a library specific for playing noises is useful. We are using the PCM library by David Mellis. This library allows us to use a playback of short audio samples. These samples are encoded directly in the Arduino sketch as an array of numbers. This allows us to code and use these audio samples in our soundboard. To download this library, use the library manager on the left side of the screen and type in PCM and install the one labelled “PCM by David Mellis” Shown below:


sketch_nov17a | Arduino IDE 2.0.2
File Edit Sketch Tools Help
Select Board
LIBRARY MANAGER
sketch_nov17a.ino
Output
Ln 1, Col 1 UTF-8 No board selected 1

LIBRARY MANAGER

pcm

Type: All
Topic: All

marc@paratte.ch>
Drive the I2C LCD Display Module LCM2004A.
jm_LCM2004A_I2C Arduino Library.
More info
1.0.2
INSTALL

PCM by David Mellis <d.mellis@bcmi-labs.cc>, Michael Smith <michael@hurts.ca>
These samples are encoded directly in the Arduino sketch as an array of numbers.
Playback of short audio samples.
More info
1.0.0 INSTALL

PCM51xx DAC by Tom Magnier
Arduino library for TI PCM51xx DAC ICs software configuration via I2C.

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

```
Firmata@2.5.8
Installing Firmata@2.5.8
Installed Firmata@2.5.8
Downloading TFT@1.0.6
TFT@1.0.6
Installing TFT@1.0.6
Installed TFT@1.0.6
```

Buzzer vs. Speaker:
Initially in our design we had been using a buzzer to produce a different tone for each button however this wasn't satisfying enough. We had each button produce a different buzzing tone, however we wanted to give each buzzer a new sound altogether. We had switched to using speakers to be able to get better quality sounds and to be able to play multiple frequencies and different tones. This tutorial will teach you how to use soundbites as arrays in your code; however if you’re looking for a simpler buzzing noise from each buzzer you can use the `tone(pin, frequency)` function. This function takes in the specified pin number associated with the buzzer and outputs a noise at the frequency given.

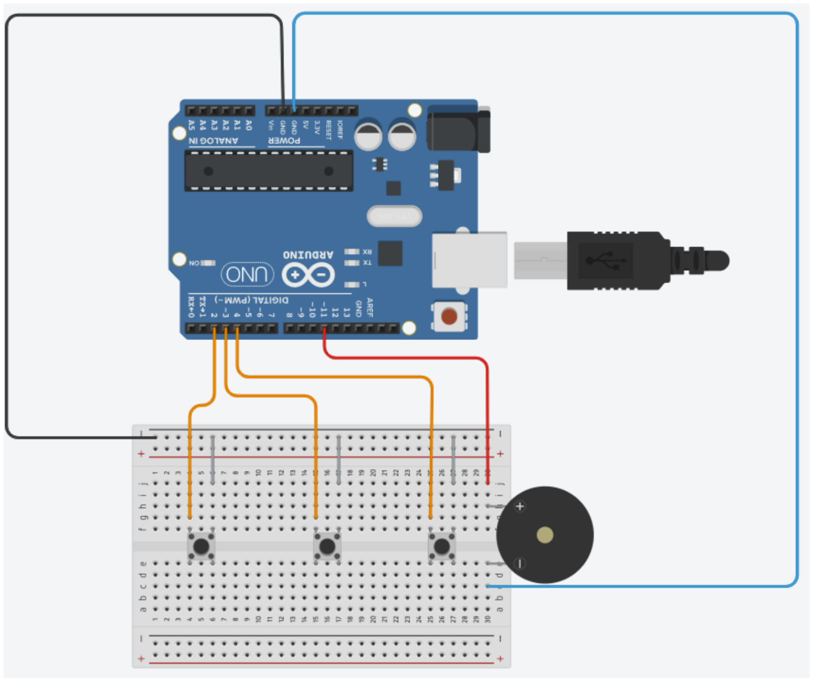
Pins of the arduino:
It's important to note that each port has a specific functionality on an arduino so Pins 0-13 of the Arduino Uno serve as digital input/output pins. These are the pins that we are using to input and output information to our arduino. More detailed information about these pins can be found here: https://www.circuito.io/blog/arduino-uno-pinout/

Materials →

- Arduino Uno
- Breadboard
- Jumper wires
- Buttons (A button for each sound)
- Buzzer or Speaker (8 Ohm 2 Watt Speaker)
- Installations: PCM library by David Mellis

Schematic →

To get started here is a schematic that can be followed to implement three different buttons to produce three sounds.



The **black** wire in this schematic connects to ground so anything along the negative track will be connected to ground, this includes the right hand side of every button which is connected to the negative track via the **grey** wires. The **blue** wire connects the negative end of the buzzer to ground as well. The other coloured wires are connecting the components to its respective port that we will be using to code information to. So the **orange** wires are connecting the buttons from left to right to ports 2, 3 and 4 respectively. The **red** wire connects the positive end of the speaker to port 11.

Code →

Once you have the sounds and their arrays acquired by using this tutorial https://www.youtube.com/watch?v=m1HEwgHSBrs&t=97s we can start on coding!

This code is using three different piano notes as three different sounds. Each is defined at the top of the code however the arrays given are extremely long so here is the drive with the code you may copy and paste from:

https://drive.google.com/drive/folders/1pUA5WsHLDjq3OncLcWUJMf41VxkDjUc6?usp=share_link

All comments about the code will be highlighted in colours to differ it from the original code.

```
#include <PCM.h> Make sure to include the library to use the startPlayback function
```

Here we define the array of the numbers that create the sound we want to use

```
const unsigned char cNote[] PROGMEM = {INSERT ARRAY};
const unsigned char dNote[] PROGMEM = {INSERT ARRAY};
const unsigned char eNote[] PROGMEM = {INSERT ARRAY};
```

Here we define the ports we want to use for each button labelled A, B and C and buzzer labelled Buzz. We will use ports 2,3,4 and 11 respectively.

```
const int A = 2;
const int B = 3;
const int C = 4;
const int Buzz = 11;
```

In arduino we always have a setup where we are initialising our pins to make sure we indicate what we want each pin to do.

```
void setup()
{
  pinMode(A, INPUT); Here we indicate we want pin A to be used as an INPUT pin, taking inputs
  digitalWrite(A,HIGH);Then we want to make sure the pin reads as a digital high when running.
                       This is repeated for Buttons B and C
  pinMode(B, INPUT);
  digitalWrite(B,HIGH);

  pinMode(C, INPUT);
  digitalWrite(C,HIGH);
}
```

Now we enter a loop, this will keep the code in this loop running until specified otherwise

```
void loop()
{

  while(digitalRead(A) == LOW)              As mentioned in the background section we are
  {                                         using a low active pin so when the button is
```

```
    startPlayback(cNote, sizeof(cNote));
    delay(750);
  }
```

pressed a LOW is output. So this checks if a low is indicated then continues with the code below it.

The `startPlayback(sample, sizeof(sample))` takes in the sample and sample size of the sound we want to play and begins to play it. We need to include a delay of size 750 ms to ensure the entire sound is being played without interruption.

This is repeated for buttons B and C.

```
  while(digitalRead(B) == LOW)
  {
    startPlayback(dNote, sizeof(dNote));
    delay(750);
  }

  while(digitalRead(C) == LOW)
  {
    startPlayback(eNote, sizeof(eNote));
    delay(750);
  }
}
```