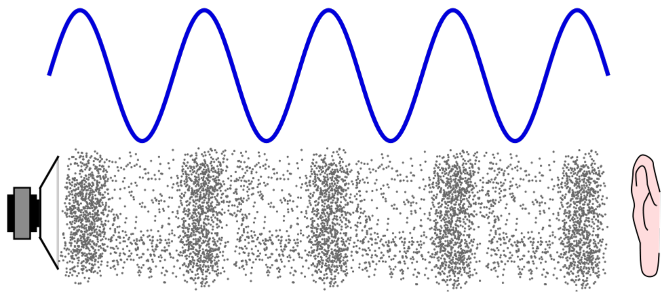
Bleeps & Blorks

As you know from science class, sounds are just air pressure waves at certain frequencies. Speakers are just diaphragms moved back and forth to create these waves. The diaphragm is moved with a voice coil, which is essentially just a coil of wire around a floating magnet. Applying a current to the coil changes the magnetic field and moves the magnet, in turn moving the speaker cone.

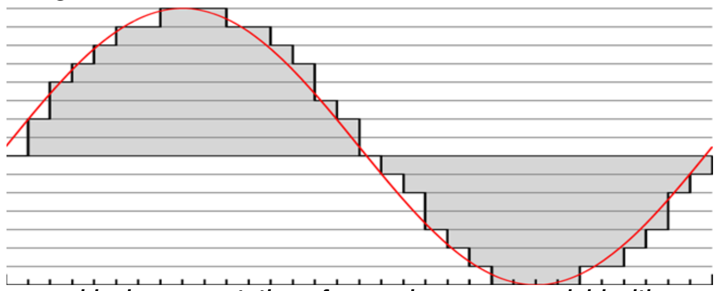
The graphic below illustrates how the frequency and amplitude look mathematically and how they actually look in free air.



Visualization of sound waves from a speaker in air, contrasted with the mathematical (analog) graph of the waveform

This means that if we can generate these pulses from our microcontroller, we can make sounds through an attached speaker! Generally, human hearing is said to be in the range of 20Hz - 20,000Hz, where Hz, or Hertz is short for 'times per second'. Hence we'll need to turn the voltage on and off at those speeds to generate the correct waves from the speaker.

Typically, audio is expressed as nice smooth waveform which is very 'analog' as shown by the red line below, but we'll just be making a square wave as shown by the grey area (remember we just have 0s and 1s, or on and off.)
But, just as your eyes can interpret a picture made up of tiny colored squares as a smooth picture, provided the blocks (pixels) are tiny enough, your ears can hear a smooth tone that's actually made up of tiny stepped changes in vibrational frequency — if they are small enough.



graphical representation of an analog curve overlaid with a corresponding digital sample

While 20,000 pulses per second sounds like a lot, our microcontroller is plenty fast enough to handle this task, since it processes instructions millions of times per second. We just need to give it the proper instructions in the proper order at just the right time.

Fortunately, someone has already done the basic work to perform the precise timing required to generate waveforms that correspond to specific notes. All we have to do is wire up a speaker, and use this existing software in our own program.

This experiment has two parts. Initially, you are going setup a basic tone generator. Then you will to evolve that into a *very* simple musical instrument.

In the first part you are simply connecting a speaker and compiling a program that will generate some tones. Next, you will connect a momentary switch, or button, that will allow you to control when the tone is played. After that you can add a second button to play another tone, and optionally, use a button to play various sequences.

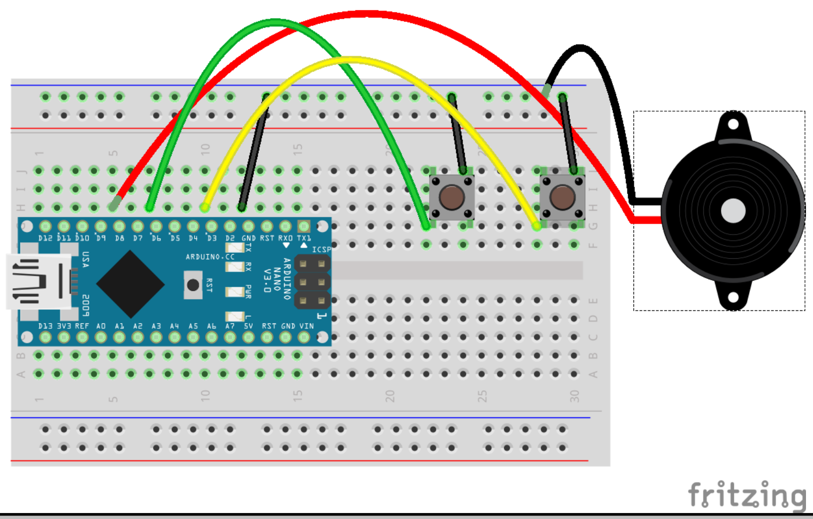
The momentary contact switch, aka button, simply completes a circuit between its two pins when pressed, and leaves an open circuit otherwise. We can use this component, along with a couple of function calls, to determine if the button is pressed and take any desired action. Continuing to check the status inside a loop, lets us take the appropriate action if it is push or not.

We can combine these various **inputs** from the buttons and the **output** to the speaker in various interesting ways.

Our setup will not be very loud; not only is the speaker small, but there is no amplification. That is, there is only so much current available to drive the speaker. A sound is considered louder when the amplitude of the waveform is larger, which moves more air, which means the speaker needs a harder push back and forth, which of course requires more (electric) power.

Consider the blue waveform in the first graphic.
There the height of the peaks and troughs represent the amplitude, make those blue curves taller, and the sound will be louder. Human hearing is not linear across the frequency range. For example, lower frequencies, or bass, requires a lot more amplitude to be perceived as being as loud as those in the middle of the range.

Experiment: Starting the digital music revolution

Components	**Wiring Diagram**
✓Computer ✓Arduino IDE ✓Speaker ✓Momentary switch (aka button) ✓Various jumper wires	

Connection Instructions

After pulling GND from the Nano to the top negative power rail, connect the speaker to ground and D8 on the Nano (try breadboard slot H, I or J5).

You can either put the speaker in the breadboard and connect jumpers, or you can just use the Male-to-Female jumper wires in your kit to keep the speaker module external, which may be more flexible.

Once you have sound, connect the buttons. Connect one wire of each button to the Ground strip, and the other side of button 1 to D6, and the other side of button 2 to D4. Be sure the wires are clear of the buttons so they will be easy to push. It can be challenging to assure that your wires and the button pins are in the same column, if you have issues, double and triple check this.

Note that it is difficult to see from the diagram, but the black wire to ground should go into the same row as one pin of the switch and the jumper back to the Arduino should go in the same plane as the other.

You can try putting something on the speaker or putting the top of the speaker on a hard surface to increase the volume, though it might alter the pitch somewhat as well.

Sketch(es)	myCasio.ino

Analysis Questions

Can you play a chord (multiple simultaneous tones)?
Why or Why not?
With 2 buttons, how many notes could your instrument play?
With 3?

Programming Tasks

Compile the sample program as is, and verify that you hear the tones. Edit the values to see if you can generate different tones.

Using what you have learned previously, add an ‘if’ and other statements so that you can press button1 and play the tone.

Now duplicate that segment such that if button2 is pushed a different tone is played.