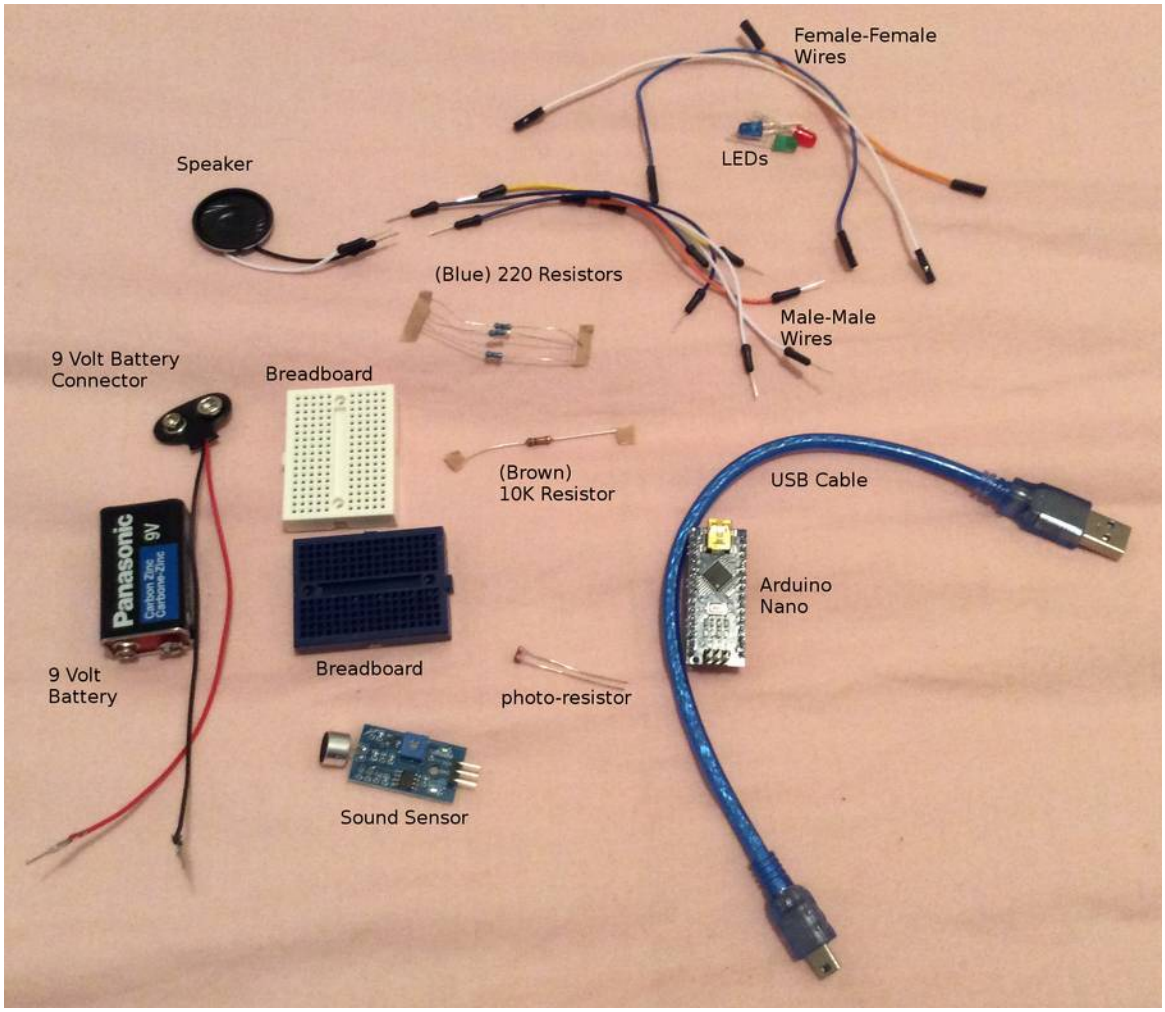
July 2019
Donna Dietz
American University

Simple Projects
for
Arduino Nano

The purpose of these projects is to give some inspiration to the study of programming. All code will be provided, but the student should read the code to gain some insight into what the code does. The purpose of this is to HAVE FUN!

What is in the kit:

Your kit should contain: One Arduino Nano and computer cable, 2 small breadboards, a 9v battery and connector, a speaker, a sound sensor (with three pins), a photo-resistor, several LEDs, some wires, at least one 220 resistor (blue), and a 10K resistor (brown).



NOTE: Lead-based solder is present in these kits. Please wash hands after use, especially prior to eating. Do not put hands/fingers in mouth after touching components.

The Code:

Download the code at https://github.com/donnadietz/nano as a zipped package and extract it on your desktop or other convenient location.

Getting Set Up:

Download an Arduino IDE for your computer at https://www.arduino.cc/en/Main/Software or http://elegoo.com/download (other sites also exist).

Ardunio code ends in an ".ino" extension. The code is stored in a folder of the same name. You can read the code on a text editor, but if you click on Arduino code after you've installed the IDE, the IDE should open it for you.

The first time around, you'll have to set things up a little. Plug in your Arduino to a USB port and try to find it using the IDE you just installed.

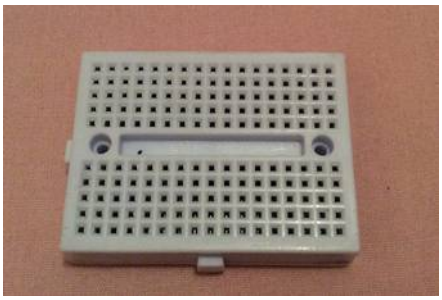
You are using an "Ardunio Nano". Under the "Tools" menu, find "Board" and choose "Ardunio Nano". Sometimes it help to try "Tools", "Get Board Info". You may need to choose the processor. "Tools", "Processor", "ATmega328P". You will have to find your own USB port if it's not automatic. For example, mine is at "Tools", "Port", and then whatever option is available at that point.

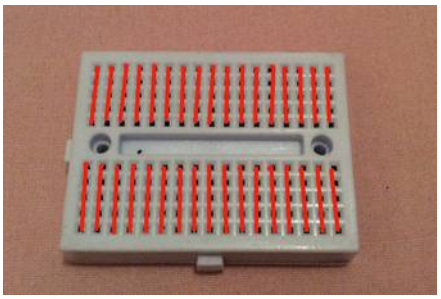
Loading the Code:

To load some code onto the Arduino, click the "right arrow" under the word Edit. If you just want to see if your code is "ok", click the "checkmark" under the word File. This tells you if the code is likely to load into the Ardunio correctly. This is useful if you're writing some code and want feedback but your Arduino is not plugged in.

How to use a Breadboard:

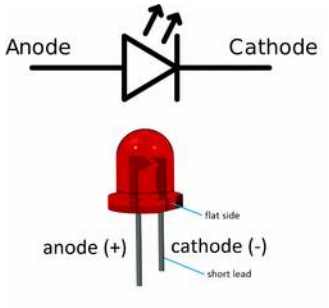
This device is called a "breadboard". It's a prototyping device that allows quick electrical connections. Inside this breadboard, there are sets of 5 holes that are connected inside as shown by red lines on the second diagram. When reading a schematic, you can think of that entire line of holes as being at one physical location. (Electrically, they are one location.)

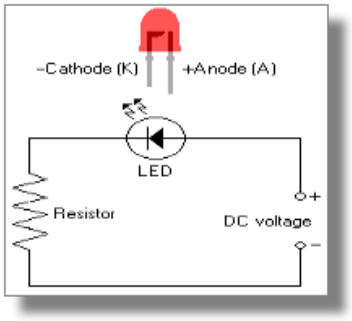


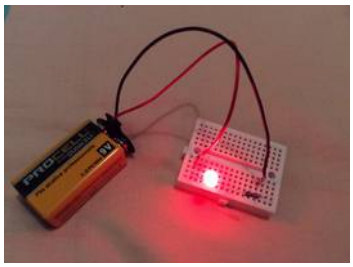


1) Test an LED without the Ardunio:

You can test an LED without using the Arduino. I grabbed some diagrams from makeyourowngear.de to help you understand LEDs better. Really, I just try them both ways to figure out which direction it will light up in. It only lights in one direction! Also, you need a 220 resistor in-line (in series) with the LED or it will either not work or it may break the LED.





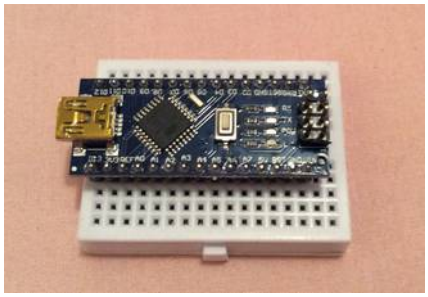


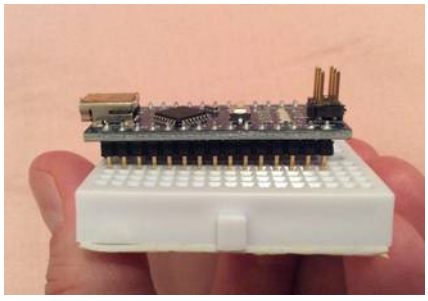
Note: the 220 resistor in this kit is blue while the 10K is brown. This is intentional. But the colors don't actually mean anything! Look up online about resistor color codes. It's the impossible-to-read color stripes on the resistors which tell you their values!

2) Test an LED with the Arduino but no code:

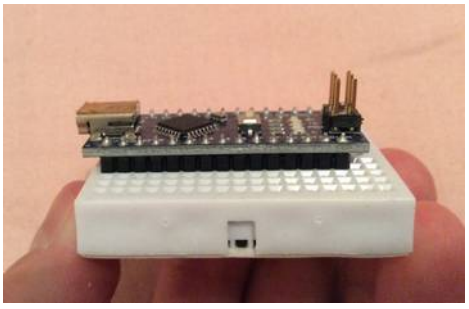
This makes sure your Arduino is taking in the battery power and passing it along correctly. You should notice that this time, the LED will not be quite as bright. It will be powered from 3.3 Volts instead of the full 9 Volts your battery can deliver.

First, attach your Nano to your breadboard. You can do this gently so the pins don't snap all the way into the board, and the connections should still work. This is best if you are a little nervous or worried about breaking things. If you are used to doing this sort of thing, you are welcome to go ahead a press the board firmly into the board, knowing that you just might bend the pins a little if you are not very graceful trying to get it back out again.

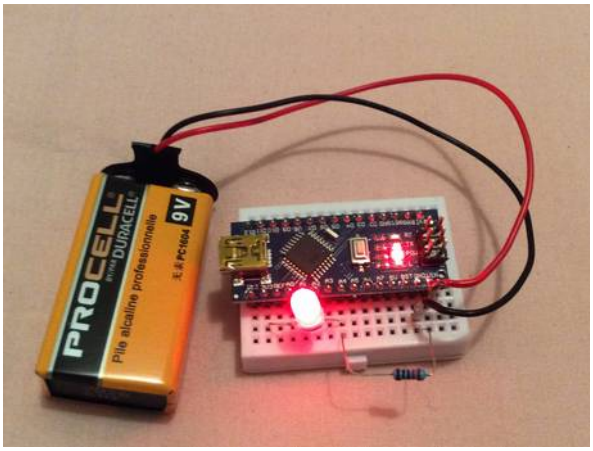




Not fully snapped in (fine!)



Fully snapped in (also fine!)



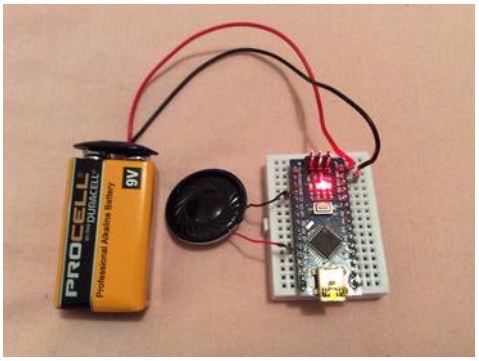
Connections:
3.3V output (3V3) to the LED long leg
LED short leg to unused location such as A3
this location to the 220K resistor
other side of resistor to GND (ground, either one)
black lead from 9V to GND
red lead from 9V to VIN

You can also move the long leg of the LED to the 5V output and see if you can notice the change in the brightness!

3) Load chimes.ino code onto Arduino to play some chimes

Open the chimes.ino file with your IDE, plug in your Arduino Nano, and hit the right arrow (upload) button. If all goes well, you should see happy results on your screen. If something goes wrong, you should see orange and red on your screen. If this happens you will have to read the errors and make adjustments. Common issues involve having the wrong board, processor, or port selected. You may have to get an updated version of the IDE in some cases, if you're using an older IDE. If you suspect it's your IDE, you can download and install a new one from another website.

Your Arduino will start executing its code as soon as it's loaded into memory. You can also restart the code by hitting the small white restart button found in the center of the board. You don't need to unplug the Arudino from the computer or attach the 9V battery unless you wish to run the code away from the computer. You can save your 9V power for those times when you wish to show off your Arudino away from the computer.



For this demo, the only thing you need to connect is the speaker. The direction doesn't matter for the speaker, but this diagram shows the black speaker wire hooked up to GND and the red wire hooked up to D8.

As before, the 9V is hooked up with red to VIN and black to GND. (Note that there are two GND or ground pins on this board.)

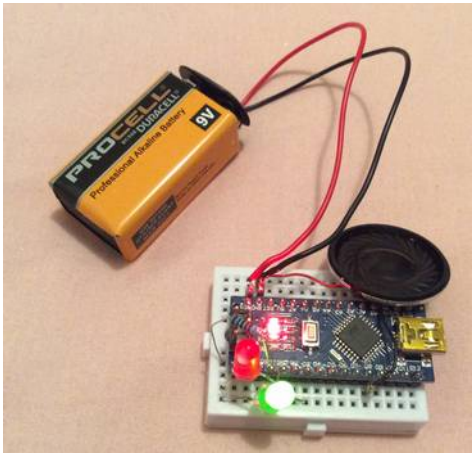
Your Arduino should start playing the Westminster Chimes melody, followed by a random number of hourly chimes (between 1 and 12).

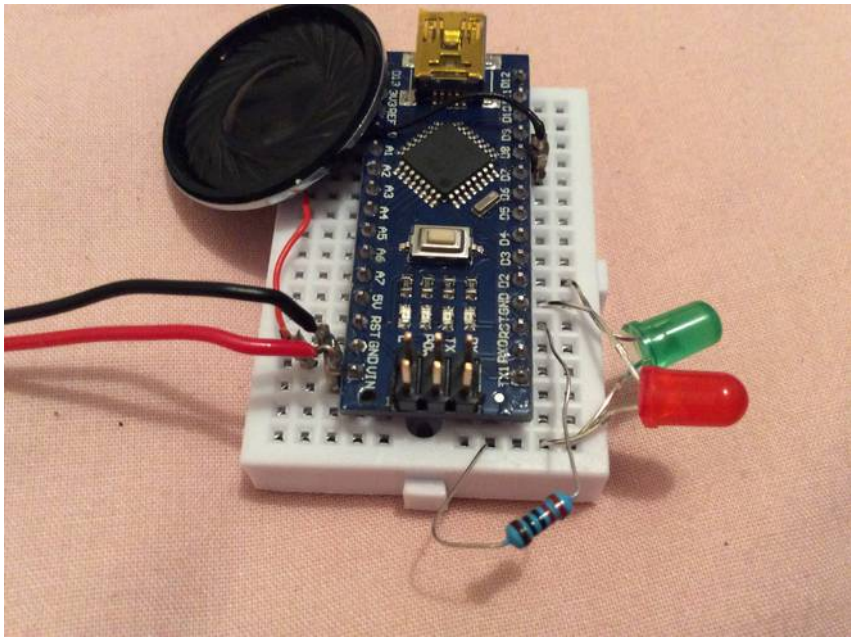
4) Load random_music.ino to play some random 5-tone music

The wiring for this demo is identical to the last one. The point to make here is to notice the difference in the code. When programming an Ardunio, you are allowed two main areas for your code, besides defining constants and libraries. The first main area is your setup() code. The second area is your loop() code. In the chimes demo, there is only setup code. In this demo, there is only loop() code. Thus, the Westminster chimes will only play once and stop. The random music just goes on until you unplug it or you run out of power.

5) Load redlight_greenlight.ino to play Red Light - Green Light

Red-light Green-light is a common childhood game. If you're not familiar with it, look it up online. In this demo, two LEDs blink and a buzzer sounds to alert you that the game is about to begin. The game randomly chooses red or green at random intervals to simulate the child who is leading the game. The tone sounds low to indicate a red light (stop) or high to indicate a green light (go).





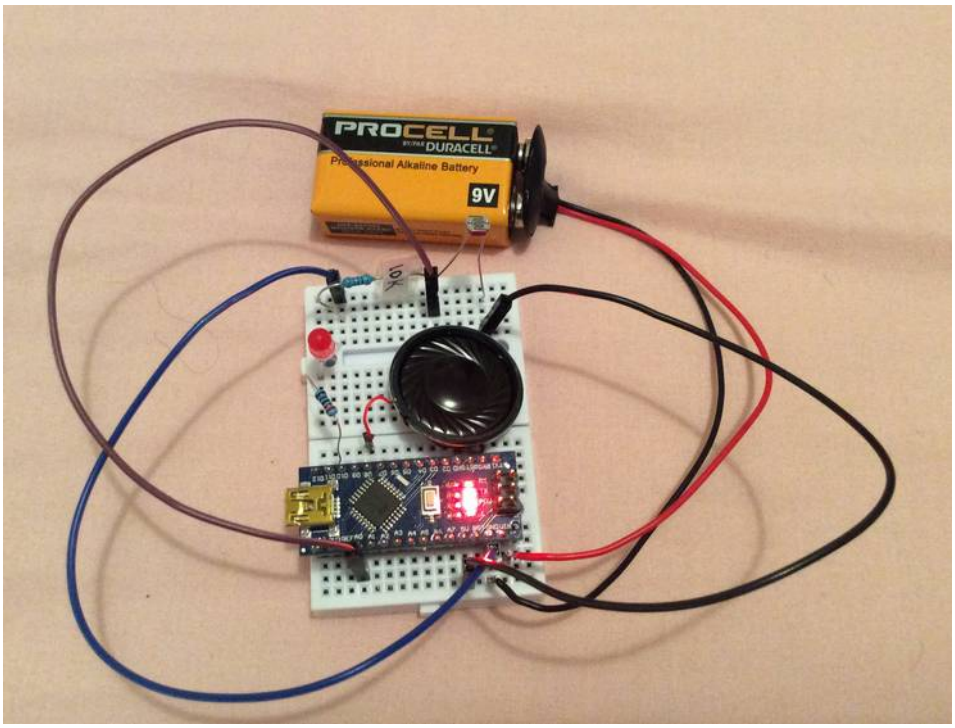
I've moved the speaker's ground connection to the other side of the board so there is more room where the LEDs are sitting.

Connections:
9V battery connected as before with red to VIN and black to GND
Speaker as before between D8 and GND
Red LED (long side) from D2 to the last set of holes on that side of the board (short side).
Green LED (long side) from D3 to the last set of holes.
220K resistor from last set of holes to GND

The resistors and the two LEDs are quite close here. Try not to let their connectors come in contact. At least try to keep the resistor away from the LEDs. If the LEDs touch a little, that should be ok. If this is difficult for you, feel free to use the other board and some of the wires to break out the connections.

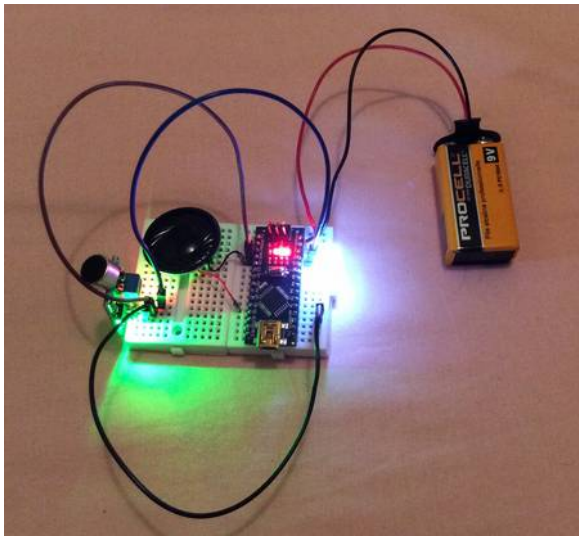
6) Load photocellprank.ino to play a prank on someone!

This demo only makes sounds when it's dark. (The code can be easily adjusted to that it only makes noises in the light or changes its sound.) It makes random noises that sound a little like an insect hiding in a drawer. The idea behind the prank is to hide the Ardunio in the dark so that as soon as someone turns on the light to figure out where the sounds are coming from, the sound stops. (Just like a real insect!) I connected my two boards together because I felt I needed the space. If you need to, you can extend the length of your male-male wires by adding a female-male to it.



Connections:
9V as usual: red to VIN, black to GND
Speaker D8 to GND
D11 to 220K resistor to LED (long side) to 10K resistor to photocell to 5V power.
At join point between photocell and 10K resistor, connect to A0.
At join point between short side of LED and 10K resistor, connect to GND.

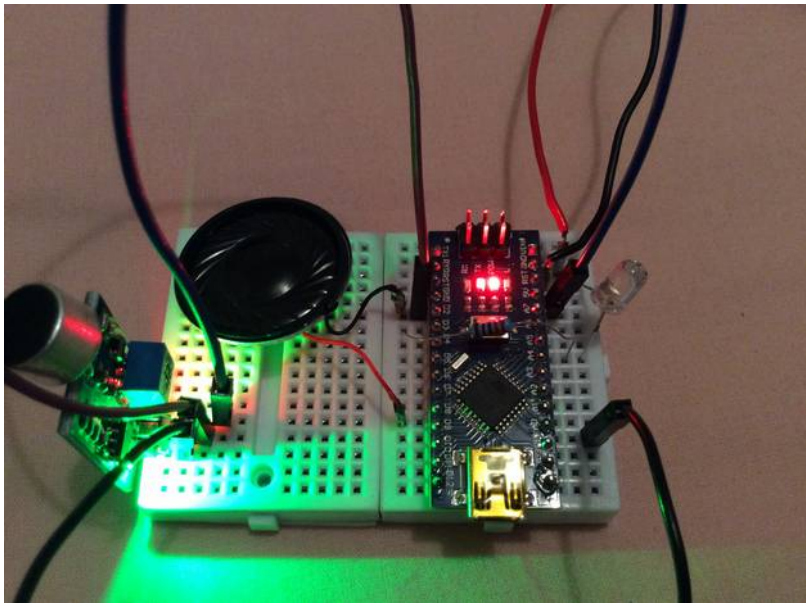
7) Load candle_blowout.ino to play Happy Birthday and blow out a candle



In this demo, the song “Happy Birthday” plays, while an LED “candle” is illuminated. When the song is done, the sound sensor should be blown into, causing the candle to flicker and eventually extinguish. (Of course, any sound will suffice, not just blowing.) You will probably need both breadboards.

You will need to add the pitches.h file to your library through your IDE for this to work. (Source: Elegoo)

Click “Sketch”, “Include Library”, then “Add .ZIP library”. Then you should be fine from there.



Set the sound sensor into the second breadboard as shown.

Connections:
The sound sensor has three labeled pins: GND, VCC, and OUT.
These should be connected to GND, 5V, and A0 respectively on the Arduino.
9V as usual, red to VIN, black to GND.
Speaker as usual to D8 and GND
LED (long side) to D2 and short side to 220K resistor* then to GND
(This is the candle, as is LED13, built-in on the board)

* Note: A5 is unused so its holes can be used as the join point for the LED and the resistor.