Brianna Martinson  and John Adkins

East Tennessee State University

Computer Science Student

Arduino Study

IDE for Arduino:

[https://www.arduino.cc/en/Main/Software](https://www.arduino.cc/en/Main/Software ) then download which ever version was made for your operating system

or

<https://create.arduino.cc/> then click on "Arduino Web Editor" and sign into your account

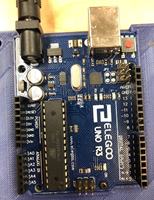
Documentation for the language used by the IDE : <https://www.arduino.cc/reference/en/>

Make sure to keep all the files on the flash drive how they are and when you make the code for the projects make sure to place the code in the correct folder, so the code can access the right libraries.

**ARDUINO**

An Arduino is an interesting tool that can teach anyone about the basics of programming and understand a circuit. In this documentation I go step by step to show how an Arduino can be programmed and how it can interact with a breadboard and the components plugged into the breadboard.

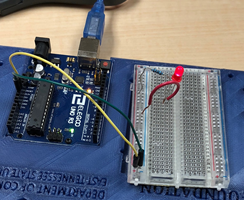
* How to give the board power from the Arduino



On the Arduino you should see small black strips with boxes cut out of them up and down the sides of the Arduino. Next to them they will have white text saying “Reset”, “3.3V”, “5V”, “GND”, and etc. on the left and “GND”, “13”, “12”, “-11”, and etc.. on the right. Take one wire and attach it to one of the black box holes on the Arduino board that has “GND” written next to it. Then plug the other side of that wire to the negative strip (blue) on the breadboard. Take another wire and plug it into the positive strip (red) on the breadboard, then plug the other side of the wire to the black box hole with “5V” written next to it. Good rule of thumb is to attach ground first and when you are taking it apart to detach ground last. The breadboard carries current up and down the positive and negative strips connects power horizontally and it also carries current horizontally on the “abcde” strips. On the letter strips is where you will place the components of your circuit e.g. LEDs.

**Light-Emitting Diode**

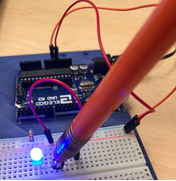
* Example circuit (Light up an LED)

**What your will need:** 3 wires, one LED, 1 bread board, 1 resistor, 1 wire to connect your board to your computer or a 9 volt battery, and 1 Arduino.

Connect your ground wire to the board and the negative strip on the board. Then take another wire and plug it into one of the black box cut that has “5V” next to it and the other side to the positive strip on the board. Next take a resistor, doesn’t matter the resistance, and plug one side into a hole in the positive strip and the other side into a hole on the “abcde” strips. To find the resistance of the resistor look at the colored bands on the resistor. Next, plug the longer lead into a hole on the same horizontal "abcde" strip as the resistor e.g. resistor end is in c8 and the longer lead of the LED is in d8. The shorter lead of the LED can be plugged into a different row of the board. Now plug in a different wire with one side horizontally next to the shorter lead of the LED and the other on the negative strip of the board. Give power to the Arduino via a 9 volt battery or your computer and if wired correctly, you LED will light up.

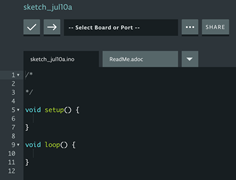
- If you would like to make your circuit more interesting. Switch out the resistor for a LDR (Light Dependent Resistor). How bright the LED will be will depend on how much light is exposed to the LDR. The more light the brighter the LED and the less light the dimmer the LED.

* Try adding in a button in the circuit to turn on and off the LED.

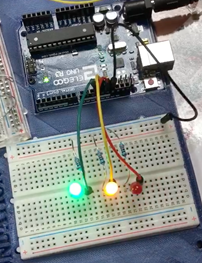
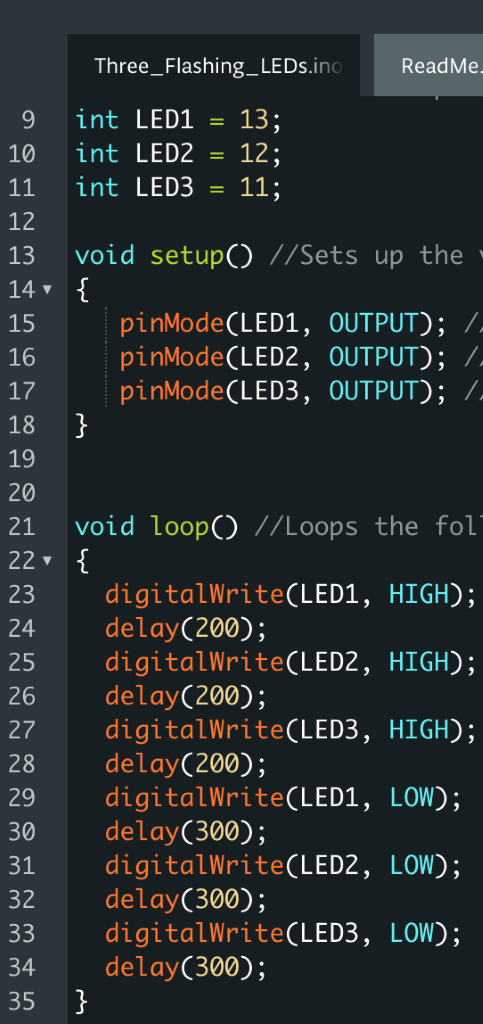


**Programming a circuit (Flashing stop light)**

**What you will need:** 3 LEDs, 3 resistors, 1 bread board, 4 wires, one wire to connect your board to your computer, and 1 Arduino.

Right now, all your board is doing is when connected to power the LED will constantly be on, but we want something more dynamic. To program the Arduino first connect your ground and your voltage but instead of plugging in the positive wire to “5V” plug it into one of the ports that has a number next to it like “13”. Connect a LED to it like before with a resistor. Now open your IDE on your computer. Before your setup function declare your LED with a variable e.g. for my board it is “int LED1 = 13;” as seen on line 3. ￼

This is just associating that port number with a name. Next, in the setup function, call the function pinMode which takes in two parameters, the port we are acting on and what we want to do with that port. In my example, I use the name of the variable we just declared, and I want it to be an output, this is seen on line 8. For more in-depth description of the pinMode method and its parameters or any other functions, use the link found at the top of this document labeled documentation. Next, in your loop function, call the digitalWrite function which takes in two parameters, the port we are acting on and the action that will happen to the port. In my example, I again use the name of the variable I used with the pinMode function and then I typed “HIGH” which turns the LED on, this is seen on line 15. Then, I called the delay function with takes in one parameter of a number that represents milliseconds, this is seen on line 16.

￼

I repeat this two more times since I am using three LEDs in my example, lines 17-20. Then I call the digitalWrite function again with the parameter of the same LED variable name, but the second variable is "LOW" to turn off the LED. Again, I call the delay function after this and then I repeat this process for my other two LEDs., lines 22-26. Now that my code is complete, I first plug in my board then I verify the code with the check mark button located below the name of your file and then I press the upload button located next to the check mark button. If all your code is right, when you click on the upload button, your LEDs should start lighting up and turning on and off. ￼

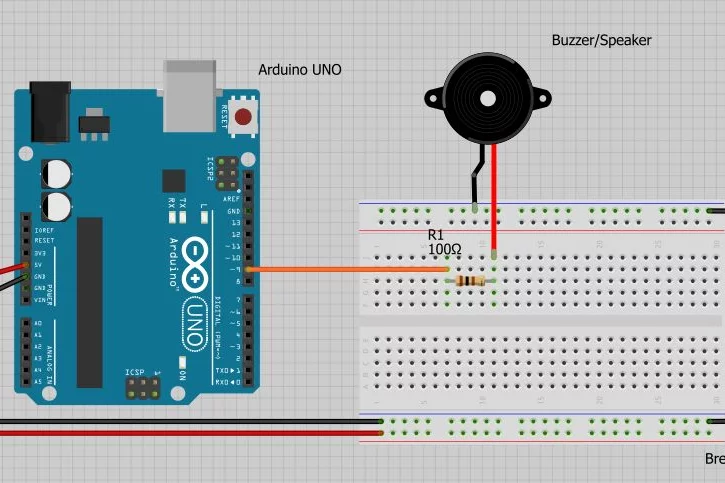
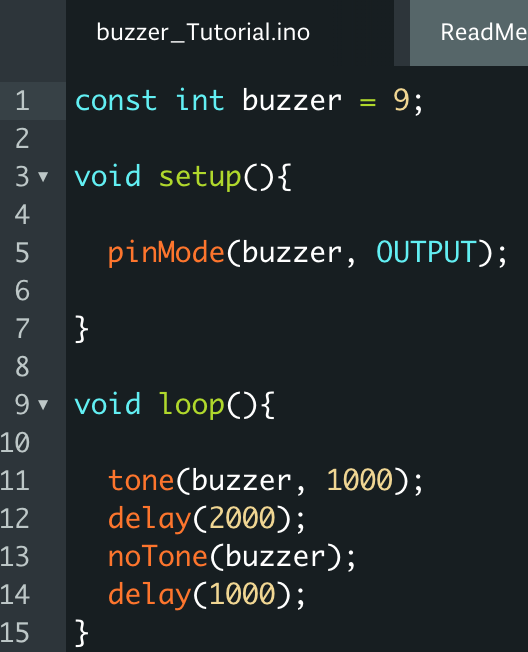
**Troubleshooting connecting board to IDE**

* Make sure you have the right item selected in the Tools > Board menu. If you have an Arduino Uno, you'll need to choose it.
* Then, check that the proper port is selected in the Tools > Serial Port menu (if your port doesn't appear, try restarting the IDE with the board connected to the computer). On the Mac, the serial port should be something like /dev/tty.usbmodem621 (for the Uno or Mega 2560). On Linux, it should be /dev/ttyACM0 or similar (for the Uno or Mega 2560) or /dev/ttyUSB0. On Windows, it will be a COM port, but you'll need to check in the Device Manager (under Ports) to see which one. If you don't seem to have a serial port for your Arduino board, see the following information about drivers.

**Buzzer Tutorial**

**What your will need: 5** wires, 1 bread board, 1 buzzer, 1 resistor, 1 wire to connect your board to your computer or a 9 volt battery, and 1 Arduino.

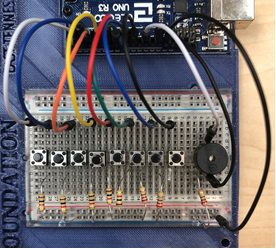
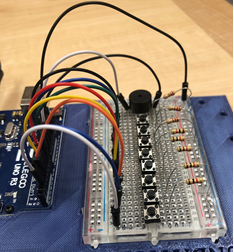
* Wire your board like the following diagram. Then open your IDE. Code like you did with the LEDs but instead interact with the buzzer. Instead of digitalWrite, you will use the method tone with the parameters of the buzzer and the frequency you want the buzzer to make. Just like before, use the delay function so then it will do the previous method for a certain amount of time. The noTone function will stop the sound from the buzzer. After you successfully get this working, try to add another buzzer. Try adding in a button to turn the buzzer on.



**Piano Tutorial**

**What you will need:** 8 buttons, 8 resistors, 1 bread board, 1 buzzer, 11 wires, 1 data cable, and 1 Arduino.

* First thing you want to do is wire all the components together. Start by lining all the buttons up in a column. Then, put a resistor for each of the buttons with one side in the negative strip of your board and the other end adjacent to the top lead of the buttons as seen in the picture. Next, you will want to put the buzzer on the board with the positive end, which is the end with the longer lead, towards the negative sides of the buttons which is the sides that the resistors are connected to. Now for the wires. Connect one wire to the negative strip that all your resistors are connected to and then plug the other end into your Arduino into a port labeled GND to establish ground or the negative charge. Now plug in a wire adjacent to the negative end of the buzzer and plug the other end into the Arduino into a slot that has a number next to it but do not use 0 or 1, for my example I plugged it into the port labeled 13. Do the same thing for the positive end of the buzzer. Now, plug in a wire to the positive side of the buzzer with will be adjacent the lead diagonal to the lead next to the resistor. Take the other end of the wire and plug it into the Arduino board in a port labeled with a number but do not use 0 or 1. Repeat this process for all the buttons. In my example, I used the numbers 2-9 for the buttons. Use the pictures below to check if your board is wired correctly.



**Code**

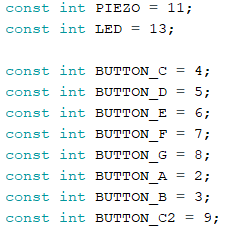
Open your IDE on your computer. Use the library in the folder. It will be in the folder. It will also contain the completed code if debugging is not working.

**Reference**



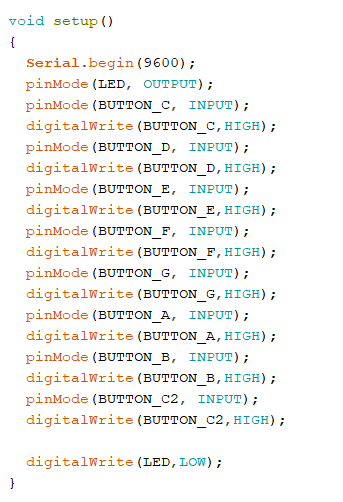
* Include the code file Notes.h. This holds the information for the different frequencies of the buzzer.

**Attributes**

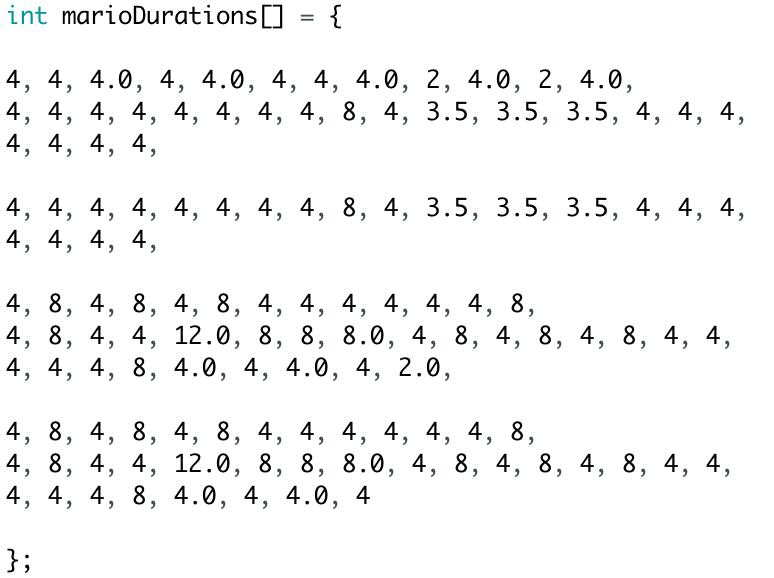
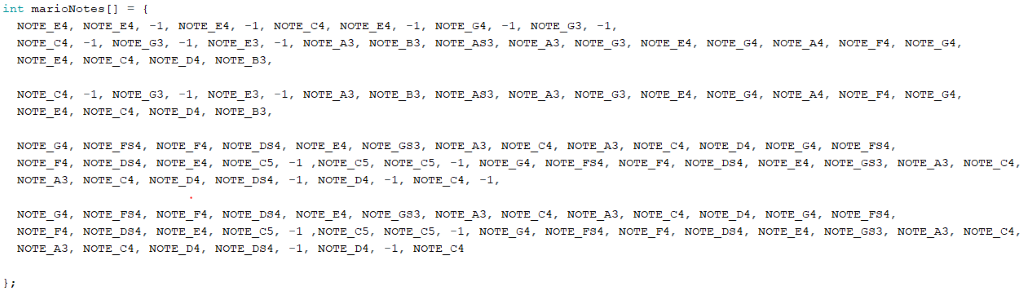


* Piezo represents the buzzer itself. LED also controls the buzzer.
* Each button represents a specific note.
* The number assigned to each variable represents the associated pin #.

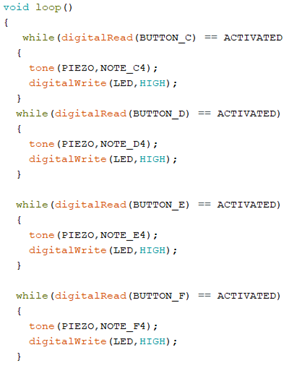
**Initialize Variables**



* Serial.begin() Sets the data rate in bits per second (baud) for serial data transmission.
* pinMode(pin, mode)
  + pin: the number of the pin whose mode you wish to set
  + mode: INPUT, OUTPUT, or INPUT\_PULLUP.
* digitalWrite(pin, value)
  + pin: the pin number
  + value: HIGH or LOW
  + If the pin has been configured as an OUTPUT with pinMode(), its voltage will be set to the corresponding value: 5V (or 3.3V on 3.3V boards) for HIGH, 0V (ground) for LOW.
  + If the pin is configured as an INPUT, digitalWrite() will enable (HIGH) or disable (LOW) the internal pullup on the input pin.

**Song**

* A song is a collection notes. An array is a collection of objects. So, these two arrays are collections of notes and time signatures that create the song.
* 4 = quarter note, 8 = eighth note, etc.



* Everything in the loop function will run while the board has power. Each of the while sections is checking to see if one of the buttons is pressed down. When a button is pressed down, it will play the note specified by the function tone being called and the parameters inside of the call. It will also call the digitalWrite function to light up the LED on the Arduino itself to show that the board is receiving the input. The if statement checks to see if someone pressed the song button. If it was pressed, it runs the code inside the brackets which loops through the two arrays that contain the notes and the notes' durations.

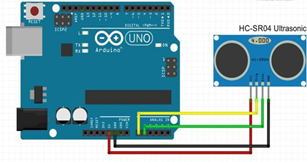


* After all the while loops and the if statement's brackets we have the code above. This code stops the buzzer's noise and turns off the LED on the Arduino, so it can be ready for the next input.

**Sensor and Servo Motor**

What you will need: 1 sensor, 4 wires, 1 Arduino, 1 wire to connect the Arduino to the computer.

* **Wiring up the Sensor**

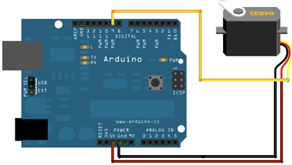


* **Code for the Sensor:** 
  + You will need to download the Ultrasound library from the folder. Then you can start coding the rest of it. Define the Sensor like you would define a LED, seen on line 8. Then declare you distance variable, seen on line 11. In your setup function you want to type Serial.begin(9600) which sets the data rate in bits per second (baud) for serial data transmission. Then in your loop function, tell the computer to read the data in centimeters, seen on line 19. Lines 21-23 will send the information from the sensor to the console. Line 25 will delay how often information gets read.

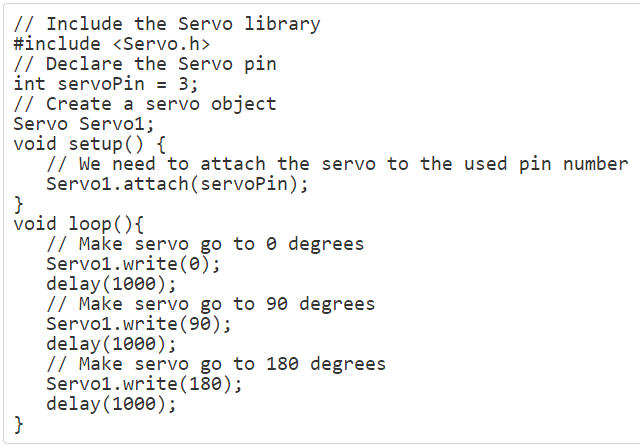


What you will need: 1 servo motor, 3 wires, 1 Arduino, 1 wire to connect the Arduino to the computer.

* **Wiring up the Servo Motor**



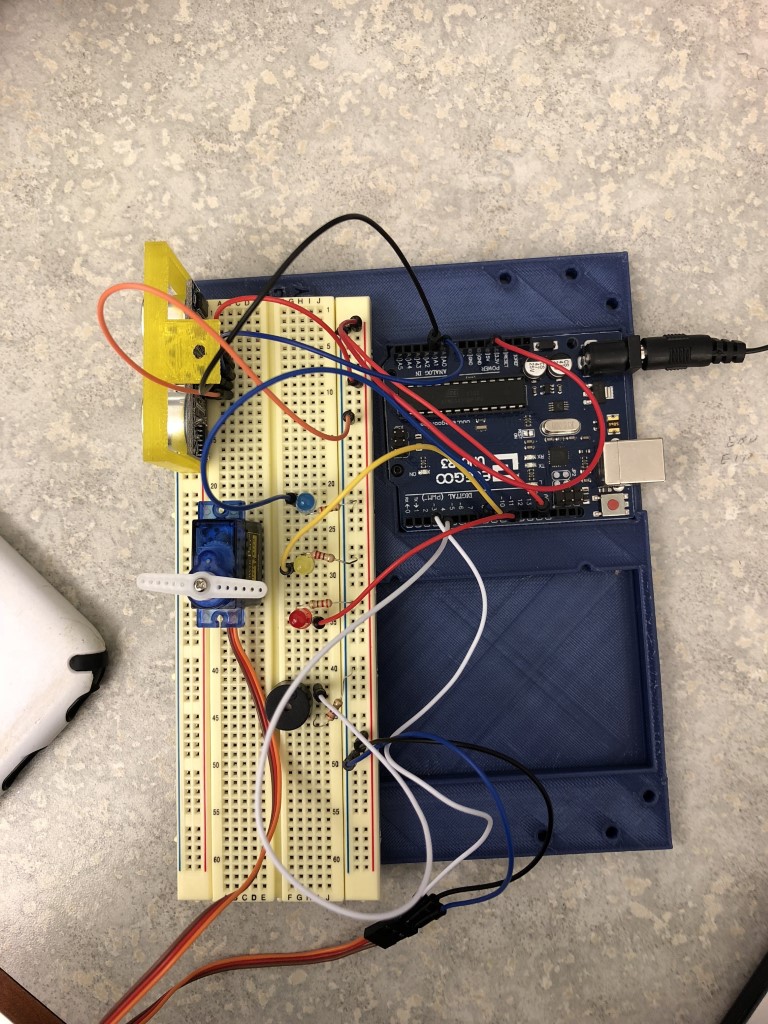
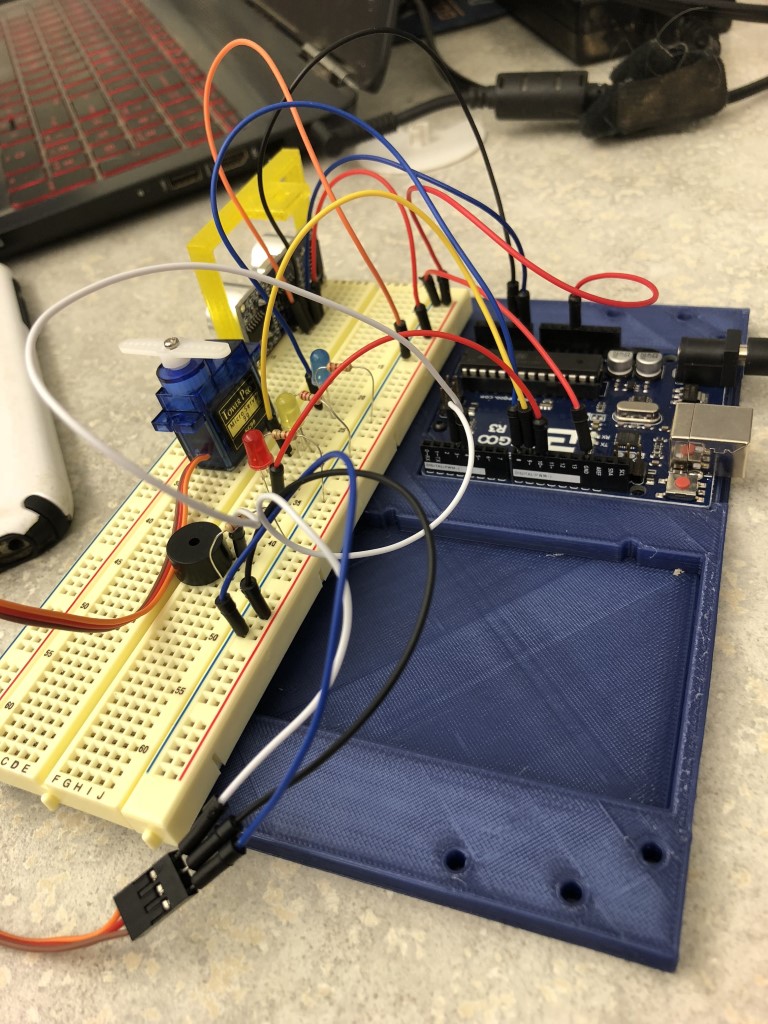
* **Code for the Servo Motor:**
  + The following code will:
  + Make the servo spin **clockwise** to its original position| wait 1000 milliseconds | then 90 degrees **counter clockwise**| wait 1000 milliseconds| then another 90 degrees **counter clockwise** | wait 1000 milliseconds. This will repeat infinitely.



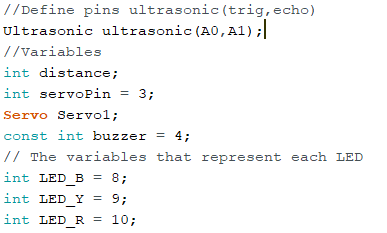
**Sound and Light Sensor**

What you will need: 1 servo motor, 1 bread board, 1 Arduino, 1 sensor, 3 LEDs, 3 resistors, 1 buzzer, 13 wires, 1 wire to connect the Arduino to the computer.

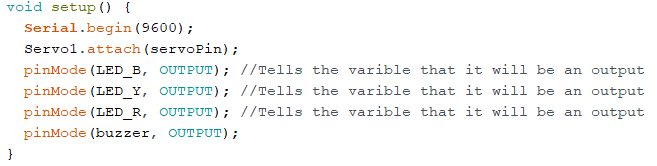
* + Wiring the sensor system:
  + Wire up three LEDs like you have before with a resistor connecting the LED to ground near the smaller lead and one wire to a port with a number from the longer lead. For my example I used 8 for the blue LED, 9 for the yellow LED, and 10 for the red LED.
  + Wire up a buzzer like you did before with a resistor grounding it from the lead without the plus sign and a wire connecting the other side of the buzzer to a port with a number from the lead near the plus sign. For my example I used port 4.
  + Wire up the sensor with a wire going from the "GND" on the sensor to the negative strip on the board, another wire going from the "Echo" on the sensor to one of the analog ports on the Arduino, another wire going from the "Trig" on the sensor to one of the analog ports on the Arduino, and another wire going from the "Vcc" on the sensor to the positive strip on the board. For my example I used analog A0 for "Echo" and A1 for "Trig".
    - Wire up the servo with the top wire, the one near the one with an arrow, going into a port with a number next to it, the next wire down going to the positive strip on the board, and the last wire going to the negative strip on the board.



* + **Code for the sensor system:**
  + Include "Ultrasonic.h"
    - Include <Servo.h>
    - Attributes



* + Setup
    - In your setup function you want to type Serial.begin(9600) which sets the data rate in bits per second (baud) for serial data transmission.
    - Attach your servo to the correct pin #
    - Set your Blue, Yellow, and Red LEDs as output and to the correct pin #
    - Set buzzer to output and the correct pin#



* + Execution Loop
    - The unit of measurement used for the distance is first established on the line right under where the loop starts. Under that, the code is telling the sensor to send the data taken in to print it to the console on the computer.
    - The delay function tells the board how often to do the tasks.
    - The three if statements check what the distance is that the sensor is reading and depending on what that distance is, it will light up a different LED and play a certain tone on the buzzer. The servo will also spin to a specific degree.
      * Less than 10 inches
        + **Blue** LED turns on, **Red** and **Yellow** LEDs turn off
        + Servo turns to 0º
        + Buzzer plays a tone with a frequency of 1000.
      * Less than 50 inches
        + **Yellow** LED turns on, **Red** and **Blue** LEDs turns off
        + Servo turns to 90º
        + Buzzer plays a tone with a frequency of 500.
      * Greater than 50 Inches
        + **Red** LED turns on, **Blue** and **Yellow** LEDs turns off
        + Servo turns to 180º
        + Buzzer plays a tone with a frequency of 200.
    - After all the if statements, noTone is called so the buzzer can be ready to play the next tone.

