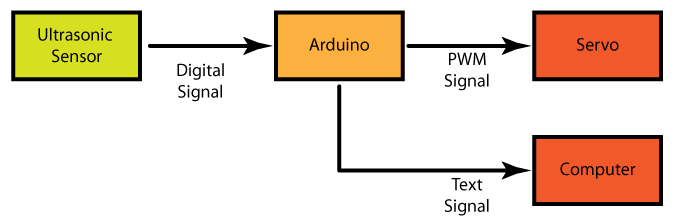
**Workshop**Weekend

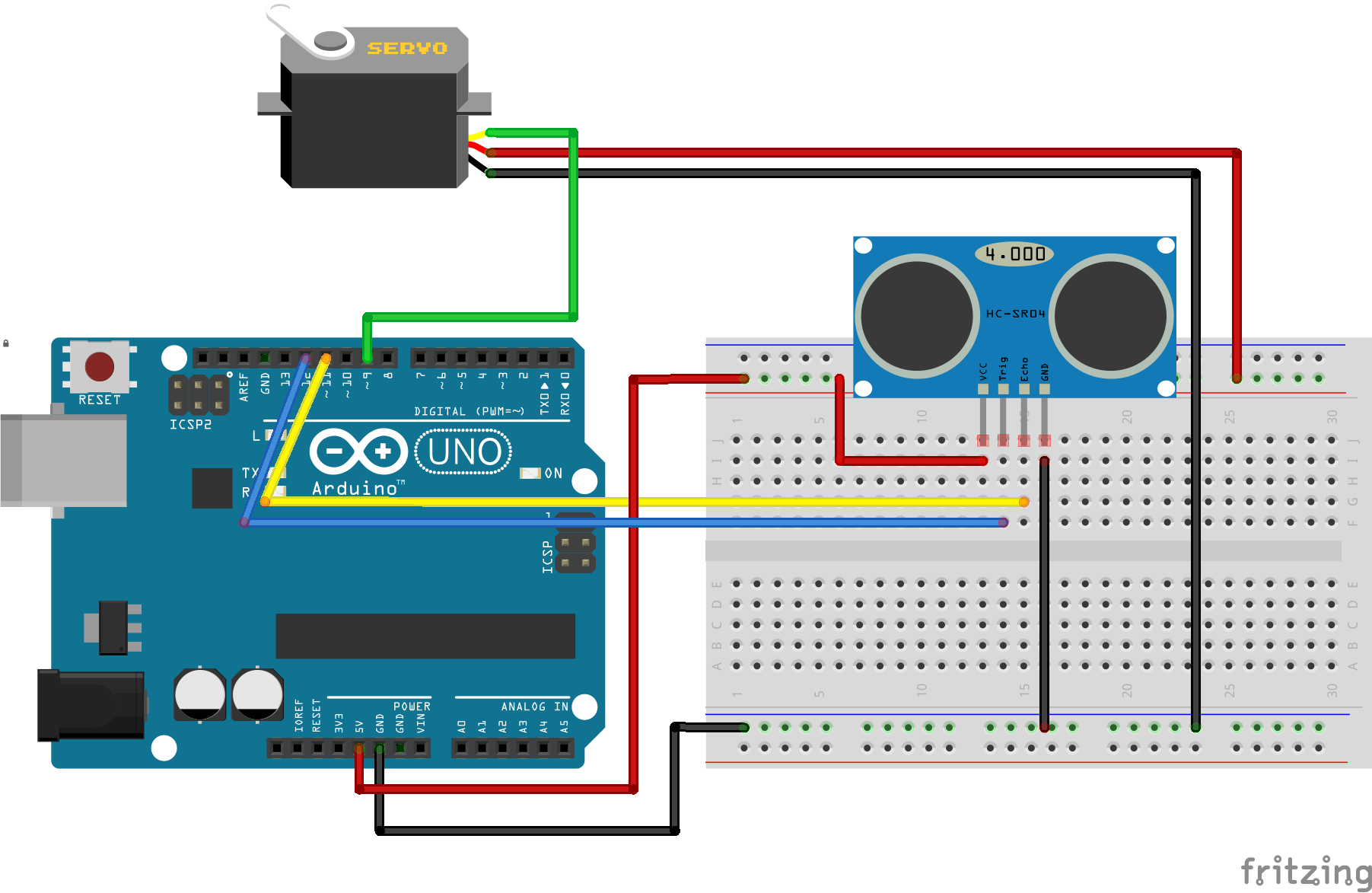
**Many Ways to Blink 4: Ultrasonic Distance Sensor to Servo**

This tutorial replaces the Light Sensor with a [Ultrasonic Distance Sensor](http://workshopweekend.net/arduino/labs/many-ways-to-blink-4). This sensor sends out high frequency sound pulses and times how long it takes for them to come back much like a bat does. We can then convert that time into a distance. The interesting thing about this sensor is that it produces a digital signal as opposed to an analog signal. This type of sensor also has a a library called the NewPing library. This library takes the digital signals coming from the ultrasonic sensor and converts them into a the duration between when the ultrasonic sensor sent a pulse and when it received it back. Working with distances is easier than working with times so the library defines conversion factors that will convert this duration into a distance. In this tutorial we will use the “US\_ROUNDTRIP\_CM” variable which is equal to 57us/cm. Since US\_ROUNDTRIP\_CM already defined by the NewPing library we do not need to define it in our code.

**Block Diagram**



**Circuit Steps**

* Unplug the Arduino. When you are change hardware the circuit should not be powered. That way you can avoid things like short circuits.
* To set up the circuit see the fritzing diagram for this tutorial 
* DO NOT plug the Arduino back in yet. It will be quite annoying if it is just running.

**Code**

* Save the “Many Ways to Blink 3” code as “Many Ways to Blink 4”. NOTE: The full code for this tutorial is at the end of this tutorial
* You can see all the chnages to the code [here](https://github.com/workshopweekend/many-ways-to-blink-4/raw/master/many-ways-to-blink-4.pdf).
* Unlike the Servo library this one does not come with the Arduino program so you will have to add it your self. You can download it from [here](https://github.com/workshopweekend/many-ways-to-blink-4/raw/master/NewPing.zip).
* Now Select Sketch —> Import Library —> Add to Library… and navigate to where you downloaded it to you.
* Restart Arduino and you should see NewPing in the File —> Examples list
* Open the NewPinExample from the Examples list
* Update the line

Serial.begin(115200); // Open serial monitor at 115200 baud to see ping results.

to

Serial.begin(9600); // Open serial monitor at 115200 baud to see ping results.

This set the Serial connection the right speed.

* NOW plug in the Arduino
* upload NewPingExample to the Arduino
* Open the Serial Monitor and confirm and distance is displayed and it changes as you move your hand closer or further away from the sensor.
* If you do not see a changing distance confirm all the hardware connections are correct.
* Switch to the “Ultrasonic to Servo” code again and include the NewPing Library like before. You can see the Tutorial 4 Code Differences pdf for details
* Now we need to initialize a Ultrasonic Sensor object so add the following to the library initialization section

NewPing sonar(triggerPin, echoPin, maxDistance);

* The triggerPin and echoPin you have seen before but the maxDistance you have not. This library requires that you define the maximum distance you will be measure. The sensor has a max distance of 400cm but for this case lets define a max distance of 200cm. So add

int maxDistance = 200; // in centimeters

to the variable initialzation section of the code.

* The big change we have is that we are no longer reading the analog pin to get our sensor value, we are reading the Ultrasonic sensor object for information so we need to change

sensorValue = analogRead(sensorPin);

to

delay(50); // Wait 50ms between pings (about 20 pings/sec).

uS = sonar.ping(); // Send ping, get ping time in microseconds (uS).

distance = uS / US\_ROUNDTRIP\_CM; // convert time to distance

The “delay(50)” guarantees that there will be enough time for the ultrasonic sensor to send and receive a signal before the next request comes. The sonar.ping() call returns?? its measurement in microseconds between when the pulse was sent out and when it was received. This is a little hard to understand so it is then converted to distance in centimeter.

* We do need to defince the “uS” and “distance” variables so lets add them to the variable initialization section. See the Code Differences for this tutorial for details. “uS” is declared as [unsigned int](http://arduino.cc/en/Reference/UnsignedInt) which like “int” declaration. If you want more detail you can read about it in the link

unsigned int uS = 0; // holds the time it took for the pulse to be received

int distance = 0; // holds the distance in centimeters

* Then we need to update the map function to reflect that we are reading the ditigal pins the ultrasonic sensor is connected to and not the analog input. So remove the lines

int sensorValue = 0; // variable to store the value coming from the sensor

and the line

int sensorPin = A0; // select the input pin for the potentiometer

* Since we are no longer using sensorValue we need to update our debugging code. So change the line

Serial.print("Sensor value: "); Serial.println(sensorValue);

to

Serial.print("Sensor value: "); Serial.println(distance);

* Then change the line

delayTime = map (sensorValue, minValue, maxValue, 200, 1023);

to

delayTime = map (distance, minValue, maxValue, 200, 1023);

* We need to update our max and min values to reflect what the new sensor will read.

int maxValue = 200; // in centimeter

int minValue = 5; // in centimeter

* The last this to do is updating the descriptions to match the hardware.
* Upload the code and open the Serial Monitor.
* Play around with different distances and see the servo blink in the same way. So we have change our input block but left the rest of the system the same. This time the interface has changes on the interface unlike when we switched to the Light Sensor. This meant we had to update the pin definitions and add the NewPing library to be able to read the sensor.
* You are done with tutorial! If you want see some more advanced topics to you can go to tutorial 5 but that is optional.

**Full Code**

/\*

Adapted From: Analog Input by David Cuartielles and Tom Igoe

Author: Malcolm Knapp

Project: Ultrasonic Sensor to Servo

Date: 4/10/14

Version: 0.1

Description: This code shows how to use a Ultrasonic Distance Sensor to

control the "blink" rate of a servo. In this case "blink" means

moving between two positions.

\*/

// ---------- included libraries ------------

#include <Servo.h>

#include <NewPing.h>

// ---------- hardware pin defines -----------

int triggerPin = 12; // select the pin for ultrasonic trigger

int echoPin = 11; // select the pin for echo

// ---------- variable initialization -----------

int delayTime = 0; //variable that holds the delay time in milliseconds

int scaling = 1;

unsigned int uS = 0; // holds the time it took for the pulse to be recived

unsigned int distance = 0; // holds the distance in centimeters

int maxValue = 200; // in centimeter

int minValue = 0; // in centimeter

int maxDistance = 200; // in centimeters

// ---------- library initialization -----------

Servo myservo; // create servo object to control a servo a maximum of eight servo objects can be created

NewPing sonar(triggerPin, echoPin, maxDistance);

void setup() {

Serial.begin(9600);

// declare hardware connections

myservo.attach(9); // attaches the servo on pin 9 to the servo object

}

void loop() {

// Input

delay(50); // Wait 50ms between pings (about 20 pings/sec). 29ms should be the shortest delay between pings.

uS = sonar.ping(); // Send ping, get ping time in microseconds (uS).

distance = uS / US\_ROUNDTRIP\_CM; // convert time to distance

// Debugging

Serial.print("uS value: "); Serial.println(uS);

Serial.print("Distance (cm): "); Serial.println(distance);

// Processing

//Scaling

delayTime = map (distance, minValue, maxValue, 200, 1023);

Serial.print ("Delay in milliseconds: "); Serial.println (delayTime);

// Modes

// None - put new modes here

// Output

myservo.write(155);

delay(delayTime);

myservo.write(30);

delay(delayTime);

}

© 2011-2016 What Will You Learn?, Inc.

[About Workshop Weekend](http://workshopweekend.net/about) | [Contact Us](mailto:support@workshopweekend.net) | [Terms and Conditions](http://workshopweekend.net/terms) | [Privacy Policy](http://workshopweekend.net/privacy)