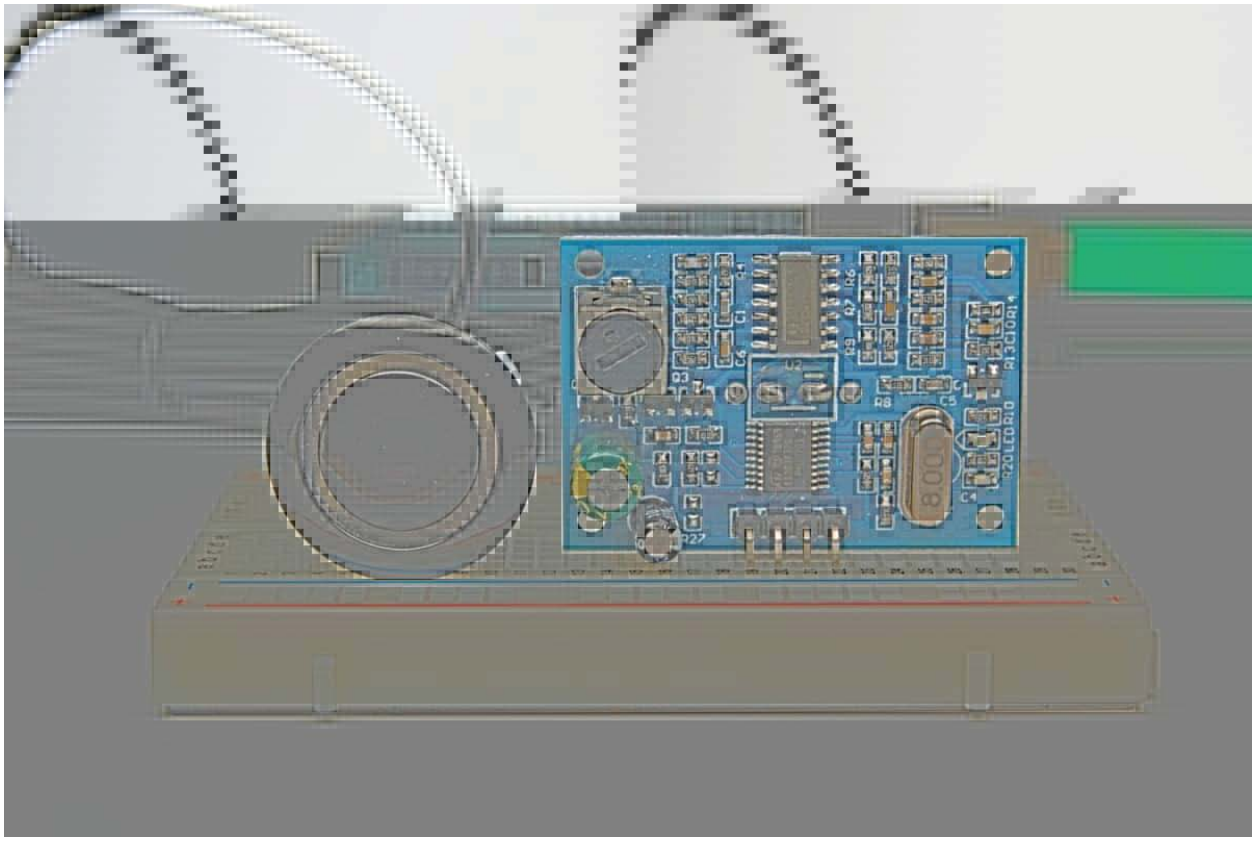
Waterproof JSN-SR04T Ultrasonic Distance Sensor with Arduino Tutorial



The JSN-SR04T is an easy to use waterproof ultrasonic distance sensor with a range of 25 to 450 cm. If you are planning to build a water level measuring system or if you need to take other distance measurements outside, then this is the sensor you need!

In this article, I have included a wiring diagram and example codes so you can start experimenting with your sensor. After each example, I break down and explain how the code works, so you should have no problems modifying it to suit your needs.

First, we will look at an example that doesn't use an Arduino library. Next, I will cover the easy to use **NewPing** library that has some nice built-in features.

Hardware components

	JSN-SR04T sensor	× 1	Amazon
	Arduino Uno Rev3	× 1	Amazon
	Breadboard	× 1	Amazon
	Jumper wires	~ 6	Amazon
	USB cable type A/B	× 1	Amazon

When shopping for this sensor, you might come across the updated version, the JSN-SR04T-2.0. This newer version works exactly the same but is rated for 3-5 V instead of 5 V. However, some users have found issues while using the sensors at a lower voltage. Using a longer trigger

puls of at least 20 µs instead of 10 µs seems to help if you are having faulty readings.

About the sensor

The sensor comes with a 2.5 m long cable that connects to a breakout board which controls the sensor and does all the processing of the signal. Note that only the sensor and the cable itself are waterproof, if you get water onto the breakout board, the sensor might stop working.

An ultrasonic distance sensor works by sending out ultrasound waves. These ultrasound waves get reflected back by an object and the ultrasonic sensor detects them. By timing how much time passed between sending and receiving the sound waves, you can calculate the distance between the sensor and an object.

Distance (cm) = Speed of sound (cm/µs) × Time (µs) / 2

Where **Time** is the time between sending and receiving the sound waves in microseconds.

So what are the differences between this sensor and the HC-SR04? The main difference, besides it being waterproof, is that this sensor uses only one ultrasonic transducer instead of two. This transducer serves as both the transmitter and the receiver of the ultrasound waves.

For more info on how ultrasonic sensors work, you can check out my article on the HC-SR04. In this article the working principles of an ultrasonic distance sensor are explained in much greater detail.

JSN-SR04T Specifications

Operating voltage	5 V

Operating current	30 mA
Quiescent current	5 mA
Frequency	40 kHz
Measuring range	25-450 cm
Resolution	2 mm
Measuring angle	45-75 degrees
Sensor dimensions	23.5 x 20 mm, 2.5 m long cable
PCB dimensions	41 x 28.5 mm

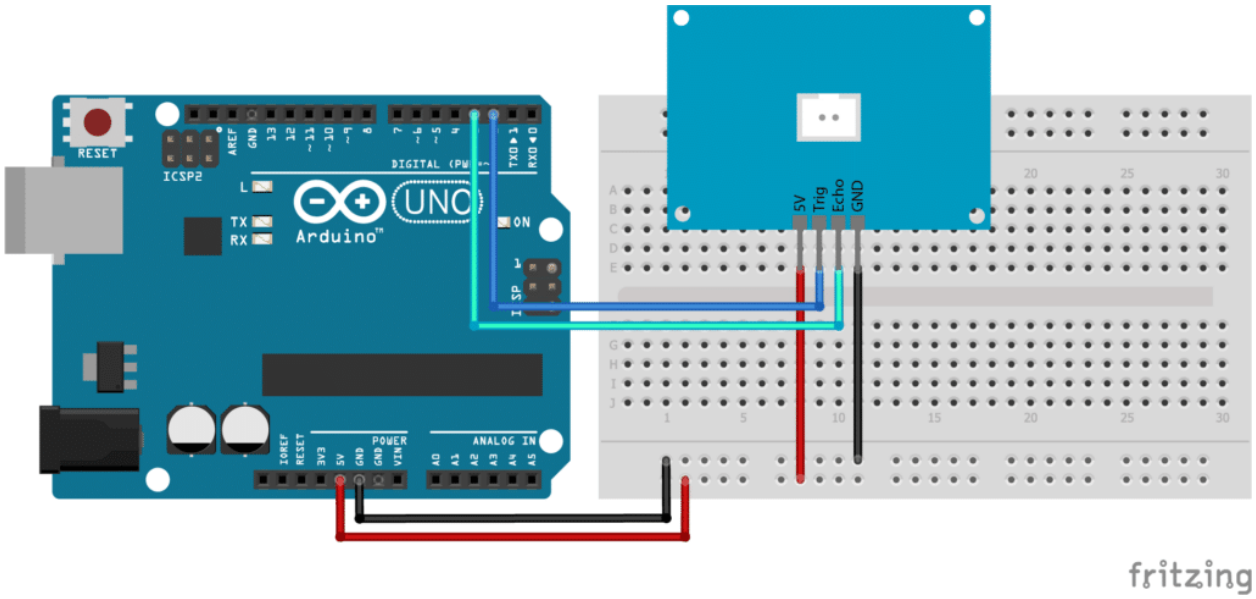
Mounting hole	18 mm

For more information you can check out the datasheet here.

https://www.makerguides.com/wp-content/uploads/2019/02/JSN-SR04T-Datasheet.pdf

Wiring – Connecting JSN-SR04T to Arduino UNO

The wiring diagram/schematic below shows you how to connect the JSN-SR04T sensor to the Arduino. The breakout board of the JSN-SR04T has the exact same pinout as the HC-SR04, so it can be used as a drop-in replacement. The cable of the sensor itself can be plugged into the connector on the back of the breakout board.



JSN-SR04T Connections

JSN-SR04T	Arduino
5 V	5 V
Trig	Pin 2
Echo	Pin 3
GND	GND

Example code for JSN-SR04T sensor with Arduino

Now that you have wired up the sensor it is time to connect the Arduino to the computer and upload some code. The sensor can be used without an Arduino library. Later I will show you an example with the NewPing library, which makes the code a lot shorter.

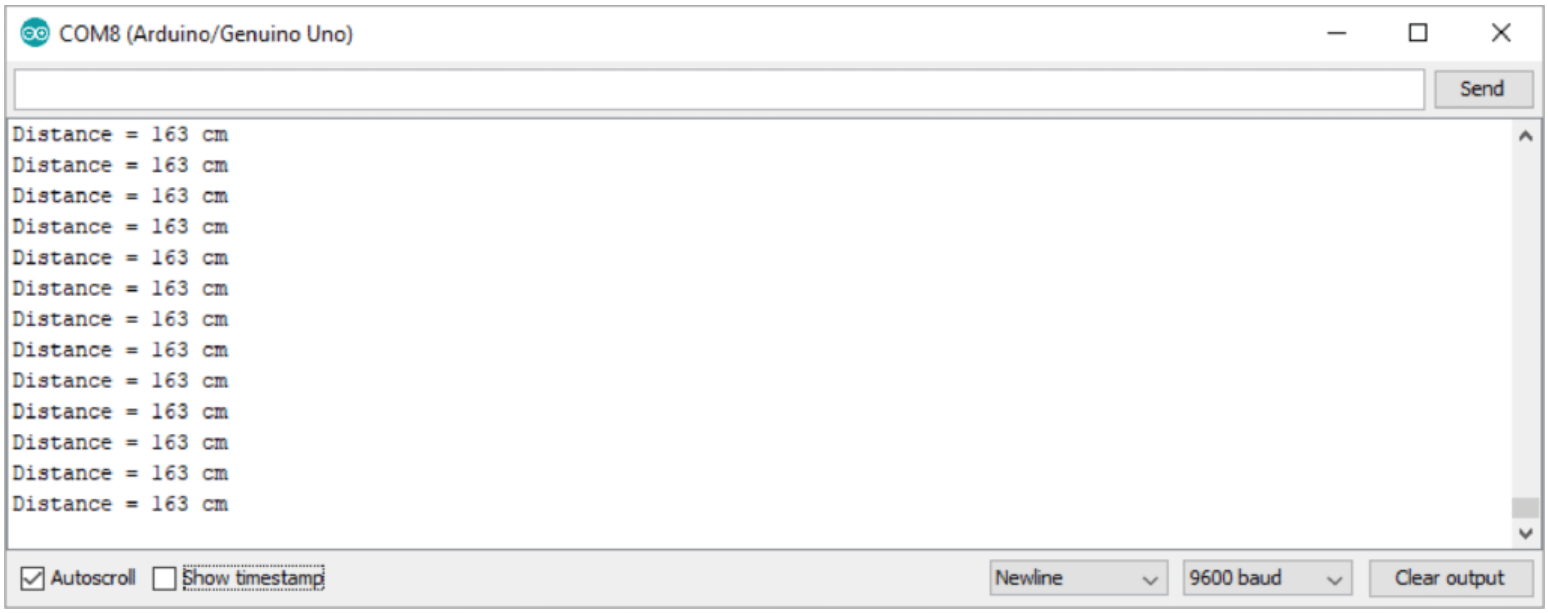
You can upload the following example code to your Arduino using the Arduino IDE. Next, I will explain to you how the code works. This code works for the JSN-SR04T-2.0 too.

You can copy the code by clicking the button in the top right corner of the code field.

```
/* Arduino example sketch to control a JSN-SR04T
ultrasonic distance sensor with Arduino. No library
needed. More info: https://www.makerguides.com */

// Define Trig and Echo pin:
#define trigPin 2
#define echoPin 3

// Define variables:
long duration;
int distance;

void setup() {
  // Define inputs and outputs
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  // Begin Serial communication at a baudrate of 9600:
  Serial.begin(9600);
}

void loop() {
  // Clear the trigPin by setting it LOW:
  digitalWrite(trigPin, LOW);

  delayMicroseconds(5);

 // Trigger the sensor by setting the trigPin high for
10 microseconds:
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read the echoPin. pulseIn() returns the duration
(length of the pulse) in microseconds:
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance:
  distance = duration*0.034/2;
```

```
  // Print the distance on the Serial Monitor
(Ctrl+Shift+M):
  Serial.print("Distance = ");
  Serial.print(distance);
  Serial.println(" cm");

  delay(100);
}
```

You should see the following output in the serial monitor:



How the code works

First, the trigger pin and the echo pin are defined. I call them `trigPin` and `EchoPin`. The trigger pin is connected to digital pin 2 and the echo pin to digital pin 3 on the Arduino.

The statement `#define` is used to give a name to a constant value. The compiler will replace any references to this constant with the defined value when the program is compiled. So everywhere you mention `trigPin`, the compiler will replace it with the value 2 when the program is compiled.

```
// Define Trig and Echo pin:
#define trigPin 2
#define echoPin 3
```

Next, I defined two variables: `duration` and `distance`. Duration stores the time between sending and receiving the sound waves. The distance variable is used to store the calculated distance.

```
//Define variables
long duration;
int distance;
```

In the `setup()`, you start by setting the trigPin as an output and the echoPin as an input. Next, you initialize serial communication at a baud rate of 9600. Later you will display the measured distance in the serial monitor, which can be accessed with Ctrl+Shift+M or Tools > Serial Monitor. Make sure the baud rate is also set to 9600 in the serial monitor.

```
void setup() {
  // Define inputs and outputs
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  // Begin Serial communication at a baudrate of 9600:
  Serial.begin(9600);
}
```

In the `loop()`, you trigger the sensor by setting the trigPin HIGH for 20 µs. Note that to get a clean signal you start by clearing the trigPin by setting it LOW for 5 microseconds.

```
void loop() {
  // Clear the trigPin by setting it LOW:
  digitalWrite(trigPin, LOW);

  delayMicroseconds(5);

 // Trigger the sensor by setting the trigPin high for
10 microseconds:
```

```
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
```

Next, you need to read the length of the pulse sent by the echoPin. I use the function `pulseIn()` for this. This function waits for the pin to go from LOW to HIGH, starts timing, then waits for the pin to go LOW and stops timing.

After that, you can calculate the distance by using the formula mentioned in the introduction of this tutorial.

```
  // Read the echoPin. pulseIn() returns the duration
(length of the pulse) in microseconds:
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance:
  distance = duration*0.034/2;
```

Finally, print the calculated distance in the serial monitor.

```
  // Print the distance on the Serial Monitor
(Ctrl+Shift+M):
  Serial.print("Distance = ");
  Serial.print(distance);
  Serial.println(" cm");

  delay(100);
}
```

Example code JSN-SR04T with Arduino and NewPing library

The **NewPing** library written by Tim Eckel can be used with many ultrasonic distance sensors. The latest version of this library can be downloaded here on bitbucket.org. You might notice that the code below, which uses the NewPing library, is a lot shorter than the code we used before.

https://www.makerguides.com/wp-content/uploads/2019/02/NewPing_v1.9.1.zip

You can install the library by going to **Sketch > Include Library > Add .ZIP Library** in the Arduino IDE.

The library does include some examples that you can use, but you will have to modify them to match your hardware setup. I have included a modified example code below that can be used with the same wiring setup as before.

```
/*Arduino example sketch to control a JSN-SR04T
ultrasonic distance sensor with NewPing libary and
Arduino. More info: https://www.makerguides.com */

// Include the library:
#include <NewPing.h>

// Define Trig and Echo pin:
#define trigPin 2
#define echoPin 3

// Define maximum distance we want to ping for (in
centimeters). Maximum sensor distance is rated at 400-
500 cm:
#define MAX_DISTANCE 400

// NewPing setup of pins and maximum distance.
NewPing sonar = NewPing(trigPin, echoPin,
MAX_DISTANCE);

void setup() {
  // Open the Serial Monitor at 9600 baudrate to see
ping results:
  Serial.begin(9600);
}

void loop() {
  // Wait 50ms between pings (about 20 pings/sec). 29ms
should be the shortest delay between pings:
```

```
  delay(50);

  // Measure distance and print to the Serial Monitor:
  Serial.print("Distance = ");
  // Send ping, get distance in cm and print result (0
= outside set distance range):
  Serial.print(sonar.ping_cm());
  Serial.println(" cm");
}
```

Conclusion

In this article, I have shown you how the JSN-SR04T ultrasonic distance sensor works and how you can use it with Arduino. I hope you found it useful and informative. If you did, **please share it with a friend** that also likes electronics!