Waterproof Ultrasonic Distance Sensors with Arduino

Waterproof Ultrasonic Sensors

Table of Contents [show]

Today we will take a look at a couple of ultrasonic distance sensors that are suitable for outdoor use.

Unlike the ultrasonic distance sensors we've used previously, these devices are waterproof.

Introduction

We have used the HC-SR04 Ultrasonic Distance Sensor many times here in the DroneBot Workshop and for many good reasons. It is pretty easy to use, it performs well, and it is very inexpensive. For collision avoidance robots or intruder detection, this sensor is usually all you'll need.

That is, as long as you are using it indoors.

However, using the HC-SR04 outside is a different story. It is a pretty fragile device and can easily be damaged by dirt, or even high winds. And, above all, it is not waterproof. Even sealing up its circuit board would not protect it, as the two transducers it uses are essentially just small speaker/microphones.

But have no fear, there are several ultrasonic distance sensors that do indeed work in harsh environments. You can get them wet, and they will still keep working, and they are also impervious to dust and dirt.

You can use these sensors to build outdoor robots, detect intruders, or serve as a backup alarm for your vehicle. And while they definitely are not as cheap as the HC-SR04, they are not all that expensive.

Let’s take a look at two of these sensors and see how they work.

Comparing Sensors

The two sensors that we are looking at today, the JSN-SRT04T and the A0YYUW, look quite different from one another. However, they both use the same principle of operation.

How Ultrasonic Distance Sensors Work

If you want a complete description of the operation of ultrasonic sensors, then please check out the article and video I did on the HC-SR04.

The operation of these sensors is summarized as follows:

Transducers

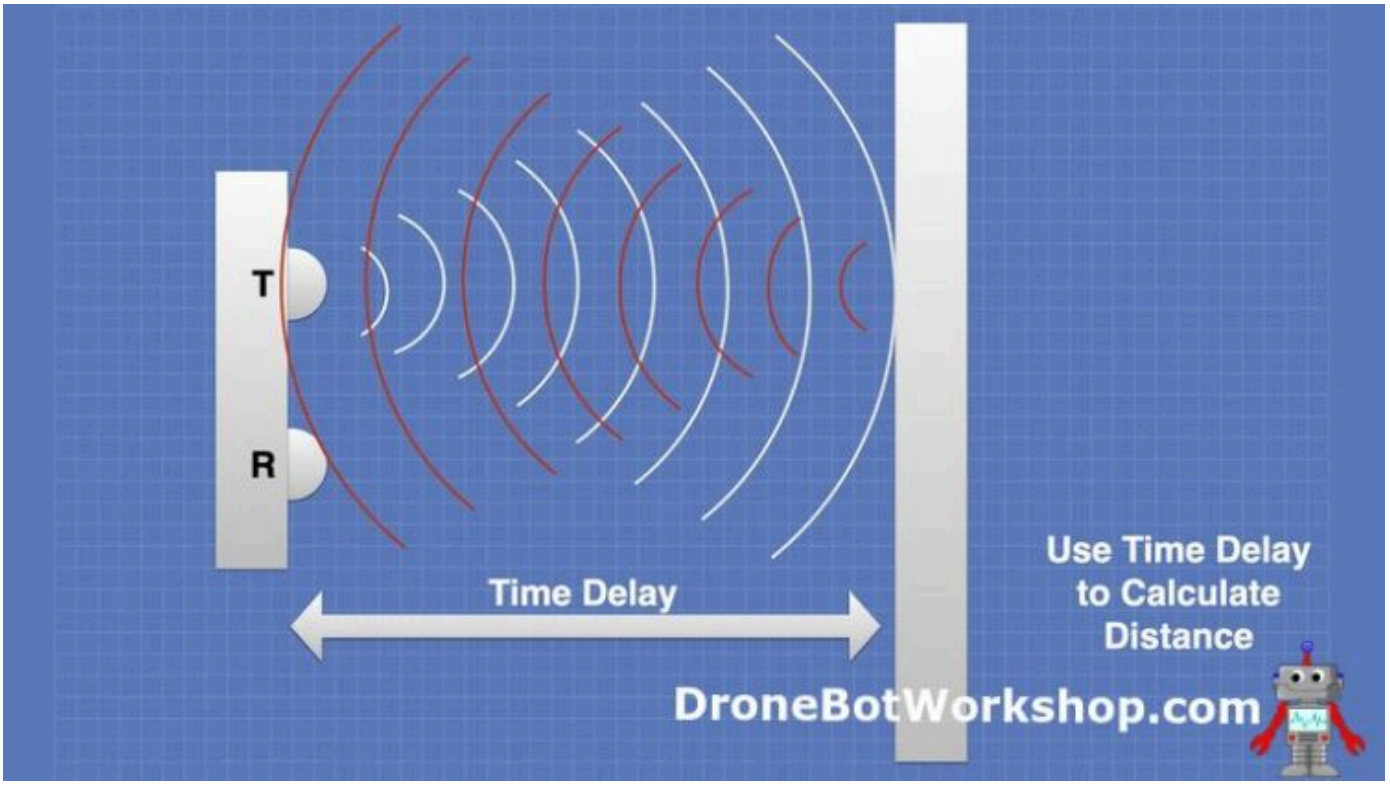
A “transducer” is a fancy name for a device that can either convert mechanical vibrations into electrical energy or convert electrical energy into mechanical vibrations.

Two common examples of transducers are microphones and speakers.

Ultrasonic distance sensors employ transducers that can work at ultrasonic frequencies, usually 40KHz. Some, like the HC-SR04 or A02YYUW, use a separate transducer to send and receive pulses of sound. Others, like the JSN-SRT04T, employ only one transducer which can serve as either a transmitter or receiver.

Ultrasonic Pulses

In operation, the sensor transmits pulses of ultrasonic sound and then listens to see if they get reflected back. If they do, then the time delay between transmission and reception is measured. This time delay can be used to compute the distance to the object that reflected the sound.



Speed of Sound

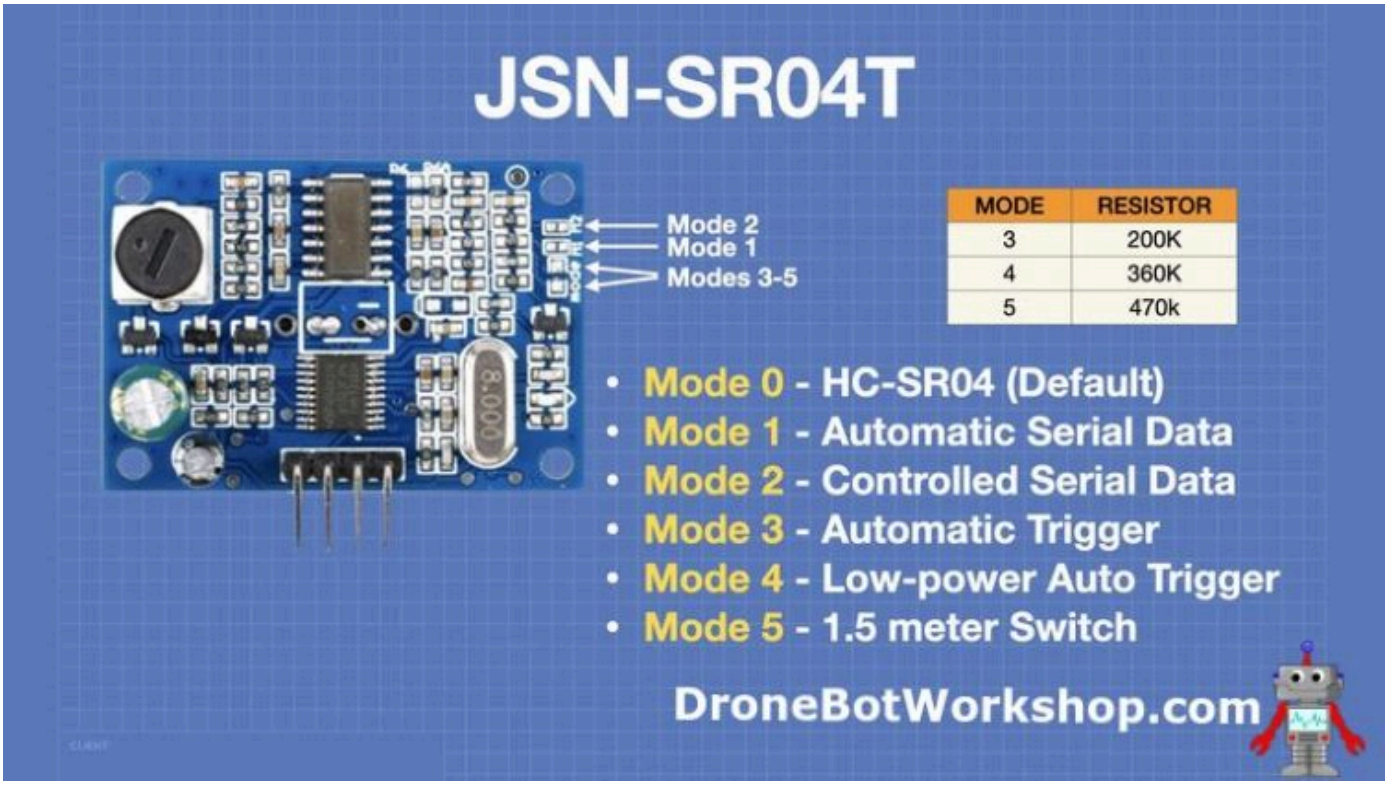
To calculate the distance, you first need to divide the time delay in half, as it represents the back-and-forth travel of the ultrasonic pulses. You then multiply the speed of sound, 343 meters per second, by the time delay to see how far the object that reflected the sound is from the transducer.

Note that the speed of sound is not a constant, it can vary due to air pressure, temperature, and humidity. It’s possible to use additional sensors to factor that in, as we did in the HC-SR04 article, but we aren’t doing that today. Of course, you could add this feature in your own sketches if you wish.

JSN-SR04T

The JSN-SR04T is an interesting sensor in that it comes as a separate printed circuit board and sensor unit. The sensor, which is a single device that acts as both transmitter and receiver, is attached to the PCB using a 2-meter cable.

Another interesting aspect of the JSN-SRT04T is that it can be run in six different operating modes.



Mode selection is accomplished in two ways:

- Mode 0 is how the sensor is configured when you buy it, no need to do anything.

- Modes 1 and 2 are selected by bridging some traces on the front of the PCB.
- Modes 3, 4, and 5 are selected by placing a resistor across some traces on the PCB. The resistor value determines the mode.

The six modes that the JSN-SRT04T can operate in are as follows:

Mode 0 – HC-SR04 Emulation

In Mode 0, which is the default mode, the sensor operates as a replacement for the popular HC-SR04 ultrasonic distance sensor. The pinout is the same, and it's essentially a drop-in replacement for the HC-SR04.

Incidentally, many people have reported that they had intermittent results using that mode and that it doesn't seem to work well with popular HC-SR04 libraries like NewPing. I have found that by increasing the length of the Trigger pulse from the common 10us to 20us, the sensor operates more reliably.

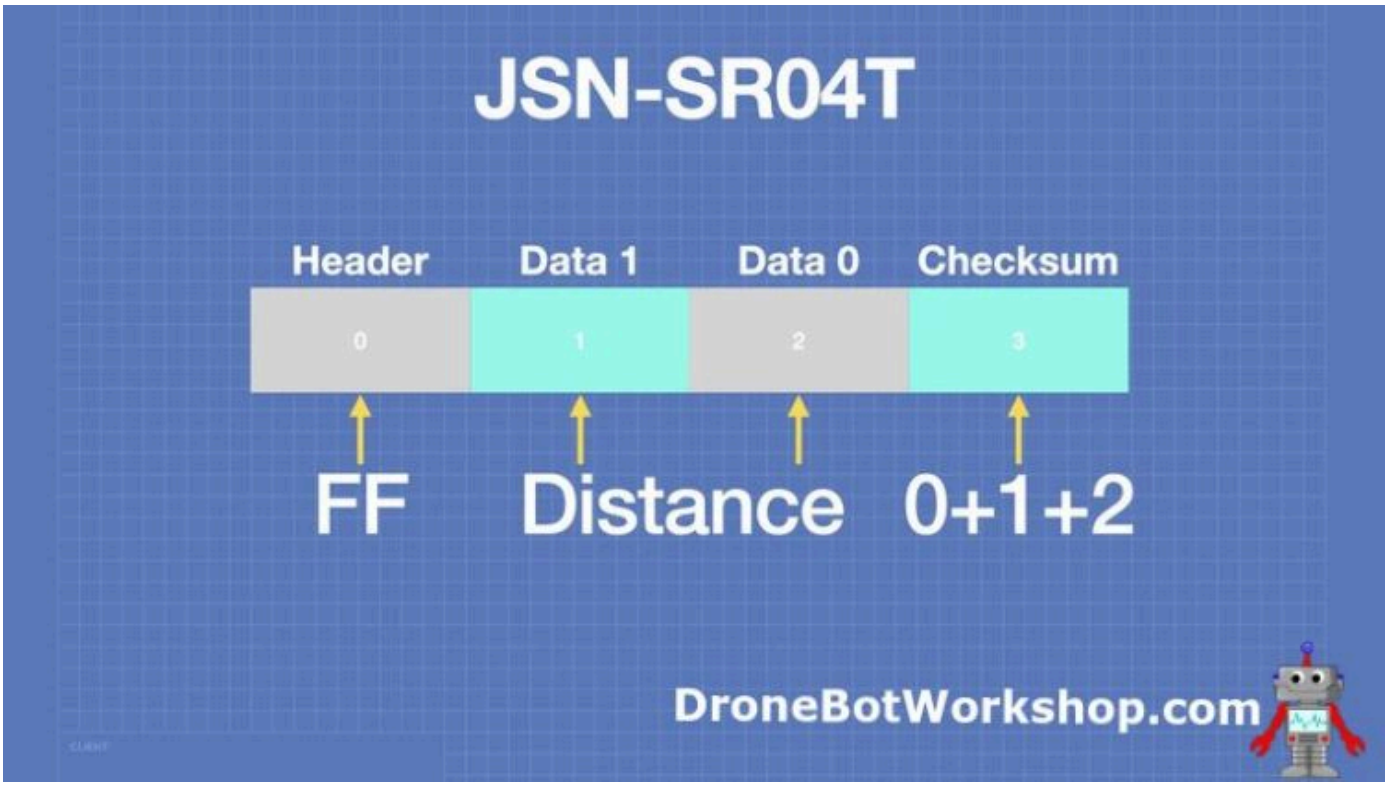
The example for Mode 0 later in this article uses this technique.

Mode 1 – Serial Data

In Mode 1 the sensor calculates the distance by itself, which is great as it alleviates this task from the microcontroller. The result is transmitted as serial data at 9600 baud (8-bits and 1 stop bit).

This mode is activated by placing a jumper or a blob of solder across the "M1" points. You can also put the sensor into Mode 1 with a 47k resistor across the "mode" points.

The format of this data is illustrated in the following illustration:



The four bytes of data sent by the sensor are as follows:

- **Byte 0 – Header** – This is always a value of hexadecimal FF, it indicates the start of a block of data.
- **Byte 1 – Data 1** – The high end of the 16-bit data, with the distance value in millimeters.
- **Byte 2 – Data 0** – The low end of the 16-bit data.
- **Byte 3 – Checksum** – The addition of the previous three bytes. Only the lower 8-bits are held here.

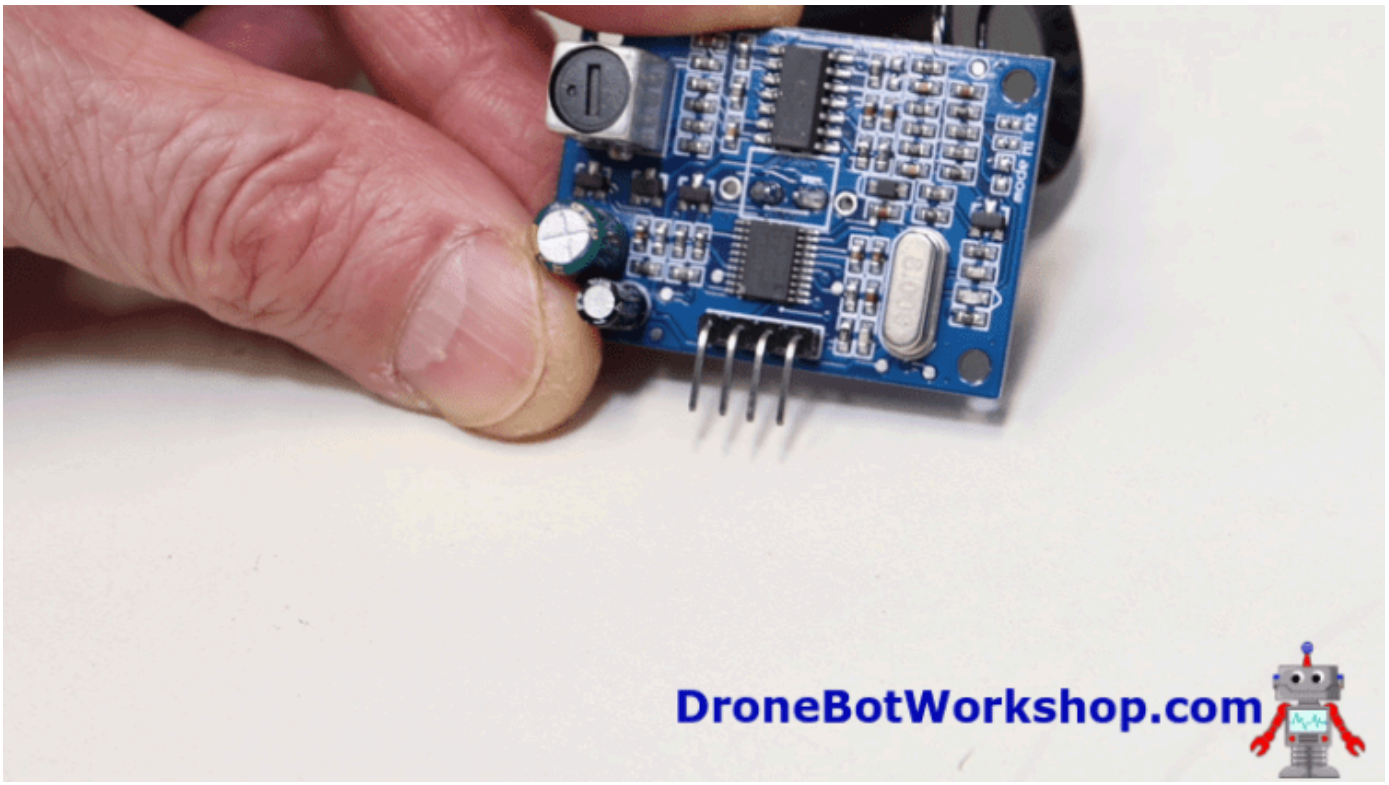
The checksum is used to validate the data, you perform the same math (D0 + D1 + D2), take the lower 8-bits, and compare it to the received checksum. If it differs, then the data is corrupt.

Mode 2 – UART Controlled Output

As with Mode 1, this mode also sends out serial data, using the same format previously described. However, in Mode 2 you need to request the data.

A data request is made by sending a hexadecimal 55 character to the sensor's RX pin. Once the sensor receives this request, it will measure the distance and send back the result.

You enter Mode 2 by placing a jumper across the M2 points, it can also be activated with a 120K resistor across the "mode" points.



Mode 3 – Automatic PWM

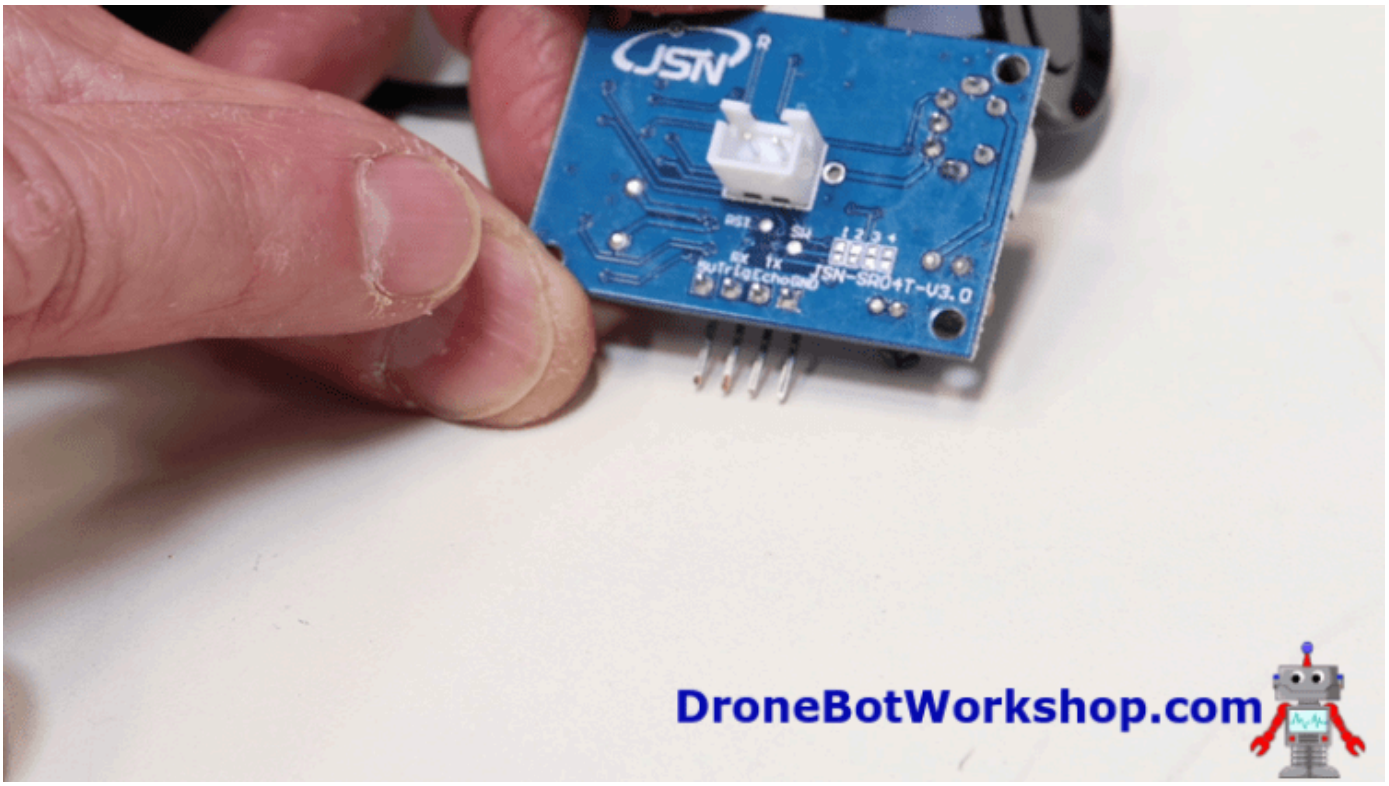
In Mode 3 the sensor acts like an HC-SR04, except there is no need to provide a trigger signal. Instead, the sensor provides its own trigger signal every 200ms. You read the data on the Echo pin in the identical fashion as you would with an HC-SR04.

Mode 3 is activated by placing a 200K resistor across the “mode” pins.

Mode 4 – Low-Power HC-SR04

In Mode 4 the sensor operates, once again, as an HC-SR04, but it uses far less current. This is accomplished by disabling the sensor’s internal watchdog timer, reducing idle current consumption to less than 70ua.

Mode 3 is activated by placing a 360K resistor across the “mode” pins.



Mode 5 – Switched Output

Mode 5 turns the JSN-SRT04T into a switch, preset at a distance of 1.5 meters. When an object is detected within this range, the Echo pin will go HIGH, otherwise, it remains LOW. No actual distance readings are returned.

This mode could be useful for intruder detection or as a backup alarm for a vehicle.

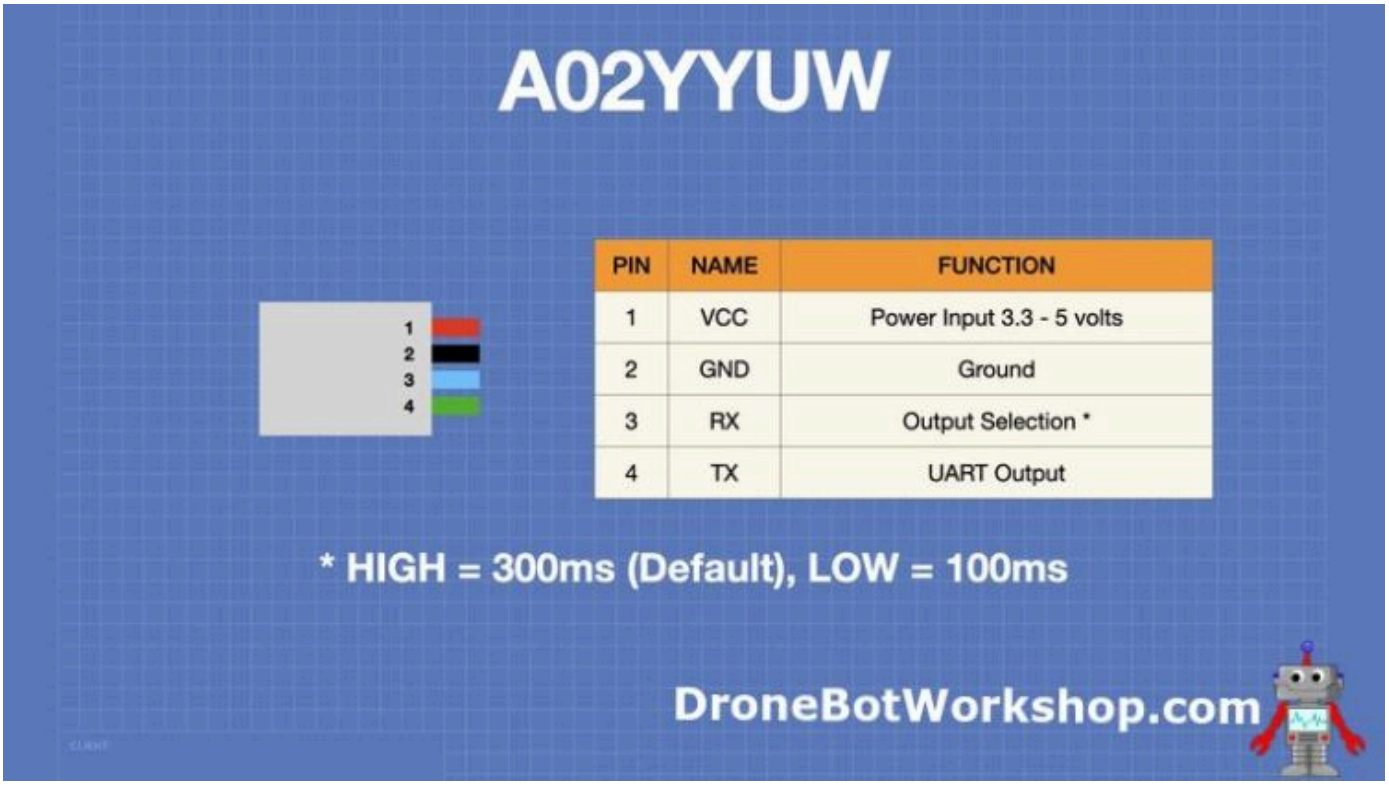
Mode 5 is activated by placing a 470K resistor across the “mode” pins.

A02YYUW

The A02YYUW is an ultrasonic distance sensor that bears some resemblance to the HC-SR04 in that it has a separate receiver and transmitter. The device is enclosed in a rubber-like material and is fully waterproof.

Despite its appearance, this sensor is not compatible with the HC-SR04. Instead, it sends serial data, in the exact same format as the JSN-SRT04T.

The device has a 4-pin connector at the end of a short cable, its pinouts are illustrated here:



Data from the device is sent using the TX pin and is at the same logic level as the power supply, which can be either 3.3 or 5 volts.

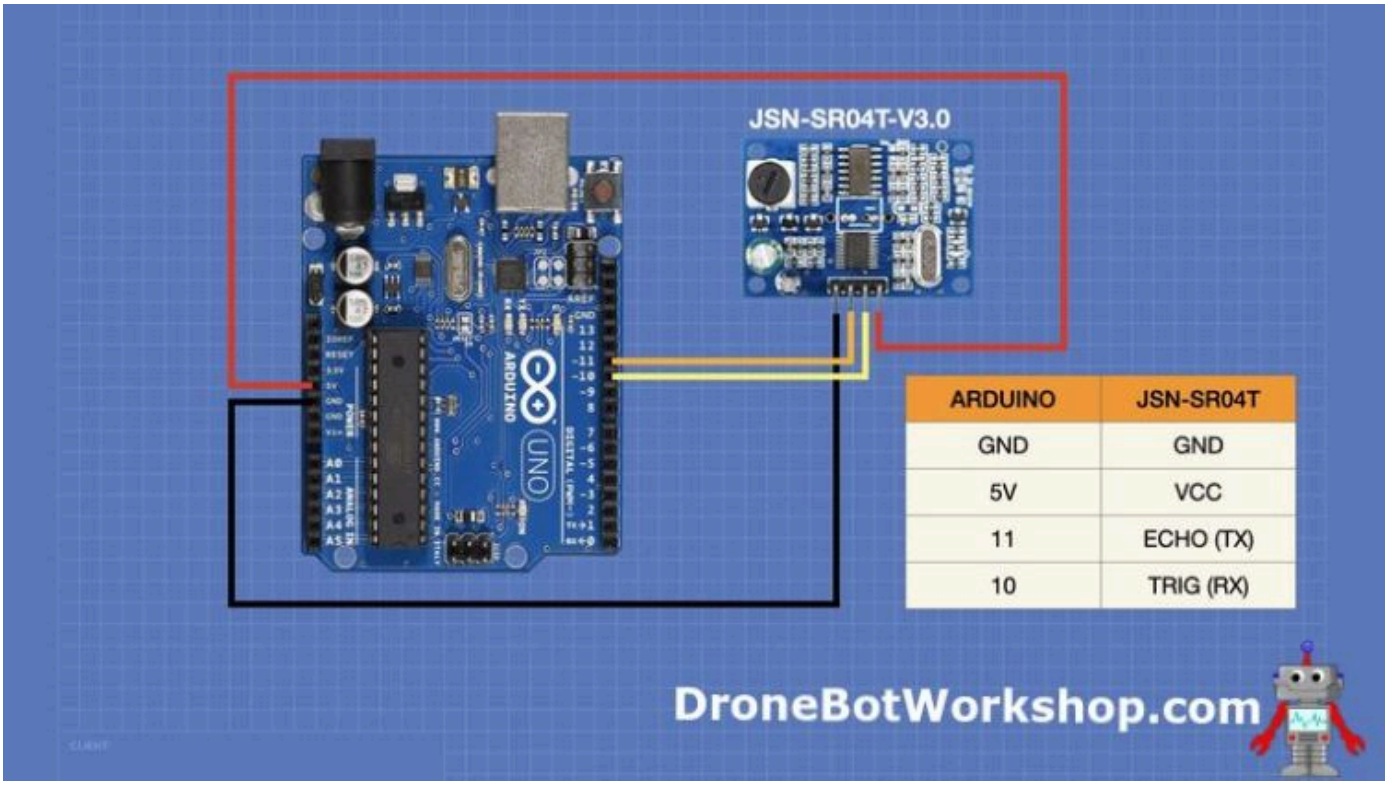
The RX pin is used to control the data output. When held HIGH or not connected (it is internally pulled up) it will operate every 300ms. If the RX pin is held LOW then the data is output every 100ms.

The 300ms rate is considered to be more stable, so unless you have a need for the data to be sent back every 100ms you're advised to tie this pin HIGH or to just leave it alone.

Using the JSN-SR04T

Now we are ready to start using our sensors. We'll begin with the JSN-SRT04T, and we will test it in both Mode 0 and Mode 1.

The hookup for both modes is identical, and is shown here:



Mode 0 Test

To put the sensor into Mode 0 you don't need to do anything to it, as this is the configuration it is shipped in.

The code we will be using is essentially identical to the code we have used before for the HC-SR04, which makes sense as Mode 0 is the mode that emulates an HC-SR04. One change though is that I've increased the pulse width of the trigger pulse to 20us, as I found it to be more stable that way.

Here is the sketch:

```
/*
  JSN-SR04T-V3.0 Ultrasonic Sensor - Mode 0 Demo
  srt04-mode0.ino
  Uses JSN-SR04T-V3.0 Ultrasonic Sensor
  Displays on Serial Monitor

  Mode 0 is default mode with no jumpers or resistors (emulates HC-SR04)
```

```
  DroneBot Workshop 2021
  https://dronebotworkshop.com
*/

// Define connections to sensor
#define TRIGPIN 11
#define ECHOPIN 10

// Floats to calculate distance
float duration, distance;

void setup() {
  // Set up serial monitor
  Serial.begin(115200);

  // Set pinmodes for sensor connections
  pinMode(ECHOPIN, INPUT);
  pinMode(TRIGPIN, OUTPUT);
}

void loop() {

  // Set the trigger pin LOW for 2uS
  digitalWrite(TRIGPIN, LOW);
  delayMicroseconds(2);

  // Set the trigger pin HIGH for 20us to send pulse
  digitalWrite(TRIGPIN, HIGH);
  delayMicroseconds(20);

  // Return the trigger pin to LOW
  digitalWrite(TRIGPIN, LOW);

  // Measure the width of the incoming pulse
  duration = pulseIn(ECHOPIN, HIGH);

  // Determine distance from duration
  // Use 343 metres per second as speed of sound
  // Divide by 1000 as we want millimeters

  distance = (duration / 2) * 0.343;
```

```
  // Print result to serial monitor
  Serial.print("distance: ");
  Serial.print(distance);
  Serial.println(" mm");

  // Delay before repeating measurement
  delay(100);
}
```

The code is pretty simple, and if you've already used the HC-SR04, you probably know how it works already.

We start by defining the connections to the Arduino. We also define a couple of floats to represent the duration of the time delay and the calculated distance.

In the Setup, we set up our serial monitor and define the pinmodes for the two connections.

We begin the Loop by bringing the trigger LOW for a couple of microseconds, then raising it HIGH for 20 microseconds, and then returning it low. This triggers the sensor to emit ultrasonic pulses.

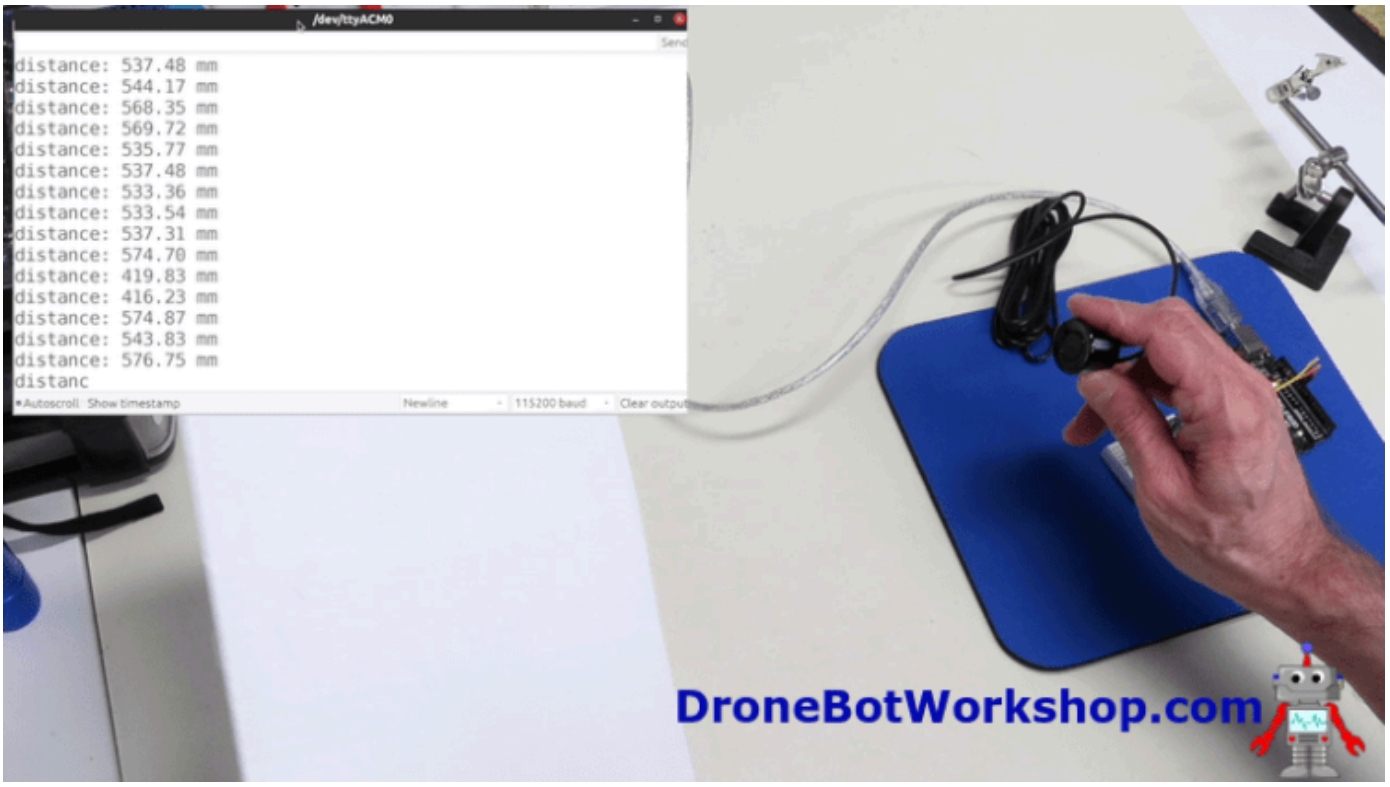
We then monitor the Echo pin to determine the pulse width of any received pulses. We accomplish this using the Arduino pulseIn function, which measures the time that the incoming pulse stays high.

We then multiply by the speed of sound, divided by 1000 as we want our results in millimeters.

Finally, we print the results to the serial monitor. Then we apply a 100ms delay to allow the sensor to stabilize and do it all over again.

Load the sketch onto your Arduino and open the serial monitor, note that the baud rate has been set at 115,200. Now observe the monitor while you move the sensor around.

One thing that you will notice when using this sensor is that once it reaches its minimum range, which is about 20 – 30 cm (200 – 300 mm), it will display false readings. At longer ranges, it works quite well.



Personally, I suspect that the limited minimum range is due to the use of a single transducer, as the device needs to be switched into receive mode after sending pulses.

Mode 1 Test

Our Mode 1 test will use the same hookup as the previous test, however, you need to set the sensor into Mode 1 first. Do this by bridging the “m1” pads on the sensor’s PCB.

Here is the sketch that we will use to read the data from the JSN-SRT04T in Mode 1:

```
/*
  JSN-SR04T-V3.0 Ultrasonic Sensor - Mode 1 Demo
  srt04-mode1.ino
  Uses JSN-SR04T-V3.0 Ultrasonic Sensor
  Displays on Serial Monitor

  Mode 1 is set by bridging "M1" pads on board

  Also works with A02YYUW Ultrasonic Sensor

  DroneBot Workshop 2021
  https://dronebotworkshop.com
*/

// Include the Software Serial library
#include <SoftwareSerial.h>
```

```
// Define connections to sensor
int pinRX = 10;
int pinTX = 11;

// Array to store incoming serial data
unsigned char data_buffer[4] = {0};

// Integer to store distance
int distance = 0;

// Variable to hold checksum
unsigned char CS;

// Object to represent software serial port
SoftwareSerial mySerial(pinRX, pinTX);

void setup() {
  // Set up serial monitor
  Serial.begin(115200);
  // Set up software serial port
  mySerial.begin(9600);
}

void loop() {

  // Run if data available
  if (mySerial.available() > 0) {

    delay(4);

    // Check for packet header character 0xff
    if (mySerial.read() == 0xff) {
      // Insert header into array
      data_buffer[0] = 0xff;
      // Read remaining 3 characters of data and insert into array
      for (int i = 1; i < 4; i++) {
        data_buffer[i] = mySerial.read();
      }

      //Compute checksum
      CS = data_buffer[0] + data_buffer[1] + data_buffer[2];
      // If checksum is valid compose distance from data
      if (data_buffer[3] == CS) {
```

```
        distance = (data_buffer[1] << 8) + data_buffer[2];
        // Print to serial monitor
        Serial.print("distance: ");
        Serial.print(distance);
        Serial.println(" mm");
      }
    }
  }
}
```

As the Arduino Uno that we are using only has one serial port, which is already in use by the USB connection, we will be using the SoftwareSerial library to create another port. SoftwareSerial is a built-in library in the Arduino IDE, so you don't need to install anything.

We define the pin connections, and also define an array with four elements. These elements will be used to hold the bytes in the data from the ultrasonic sensor.

We also define variables to hold the distance and the checksum value, and we create an object to represent our serial port.

In the Setup, we initialize both the serial port and the SoftwareSerial port.

Now we move into the Loop, where we listen to see if we are receiving data on our SoftwareSerial port.

If we detect data, then we apply a small delay to allow it to stabilize. We then look for the hexadecimal FF, which indicates the header byte.

When the header byte is found, we store it in the first element of the array. We then continue to store the next three bytes in the remaining array elements.

Now that we have a full packet of data, we compute the checksum by adding the first three array elements together and comparing the result to the fourth element, which carries the checksum calculated by the ultrasonic sensor. If these match, then the data is good.

Once we have good data, we join the second and third array elements to get the value, which is then printed to the serial monitor.

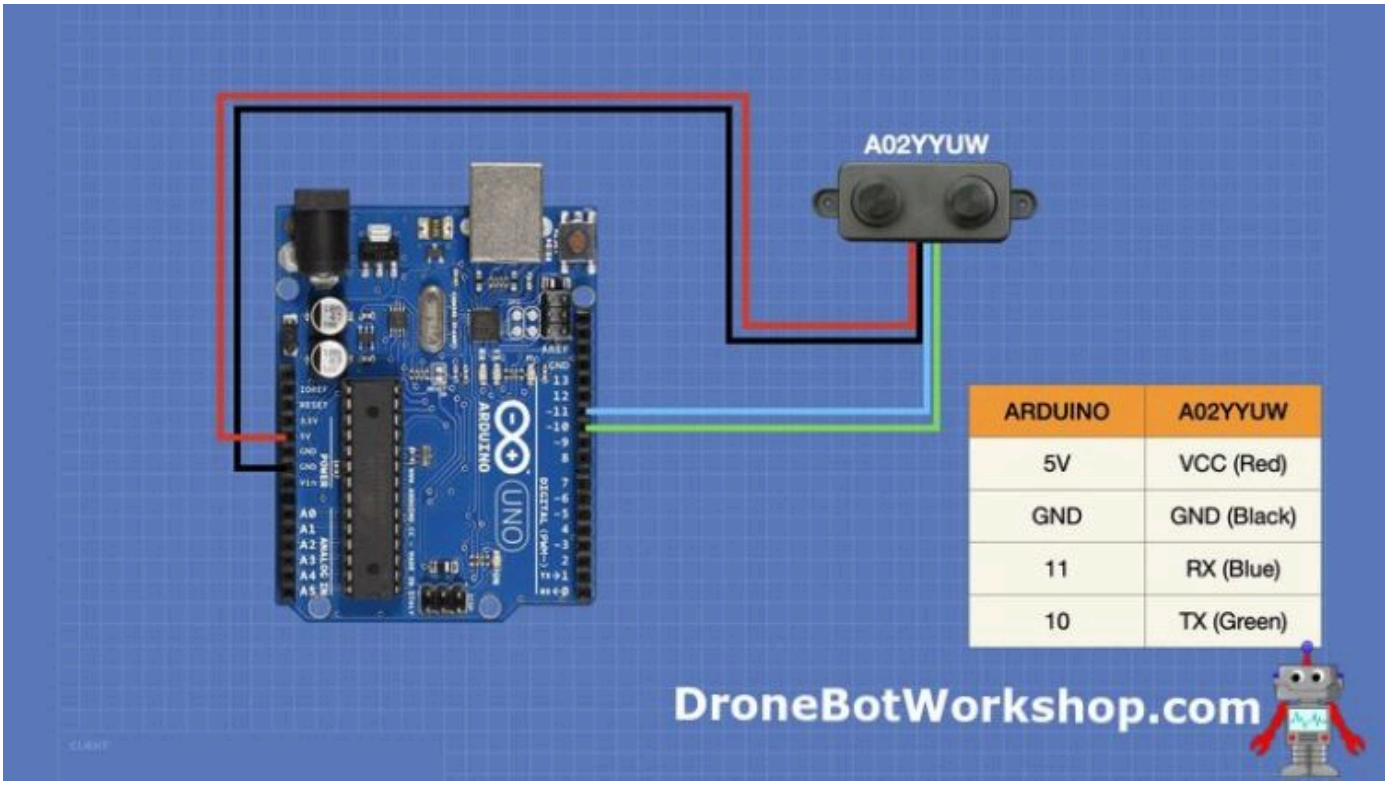
Load the sketch and observe the results. They should be similar to the results you obtained using the JSN-SRT04T in Mode 1.

Using the A02YYUW

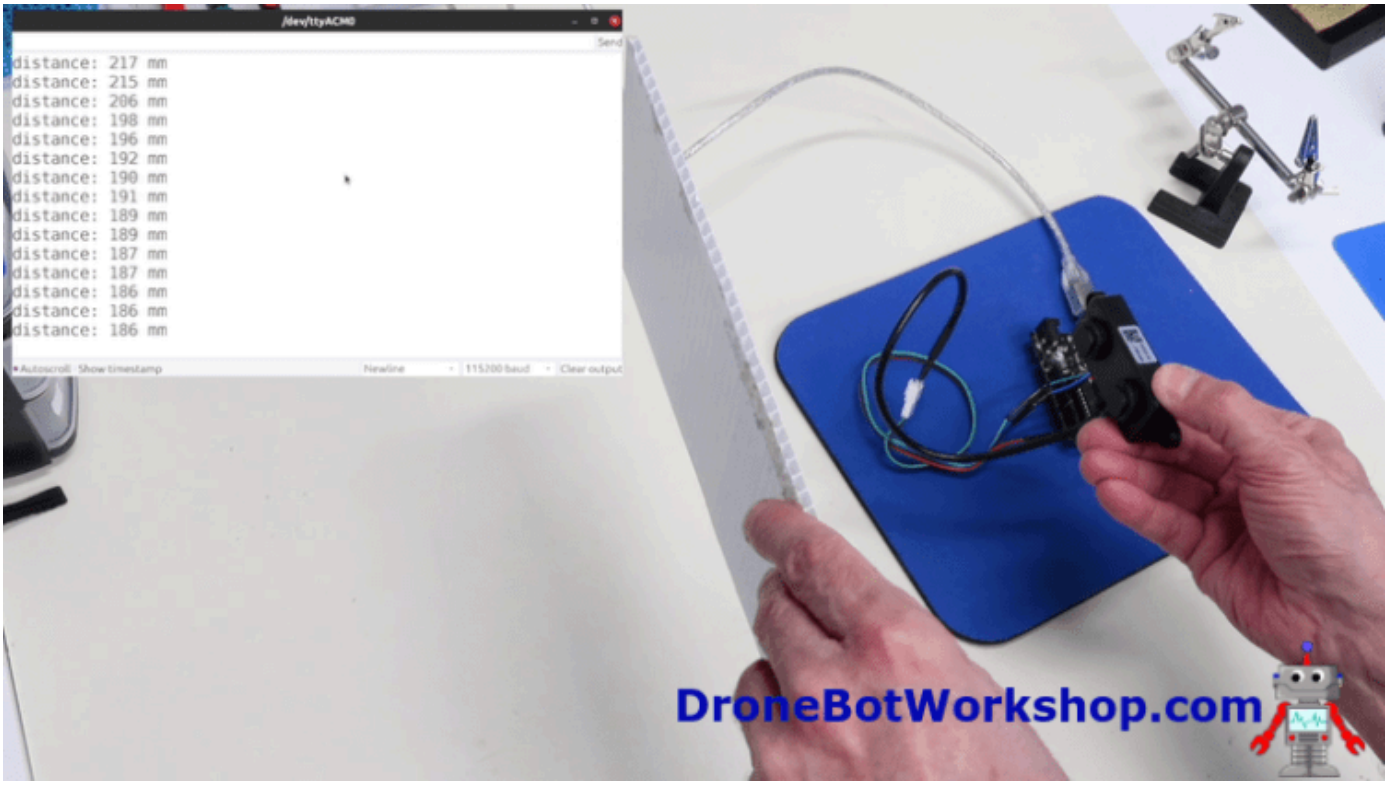
Now we will move on to our next ultrasonic distance sensor, the A02YYUW.

This sensor only outputs serial data, and it uses the exact same format as the JSN-SRT04T in Mode 1. So after we hook it up, we can use the previous sketch to read the data.

Here is the hookup of the A02YYUW with an Arduino Uno:



As you can see we have used the same connections for the A02YYUW as we did for the JSN-SRT04T. So we can use the sensor with the Mode 1 sketch above without alteration.



Once you have it hooked up and the sketch loaded, give it a test. One thing that you'll immediately notice is that it can be used at a much closer range than the JST-SRT04T, it's good down to about 3 cm.

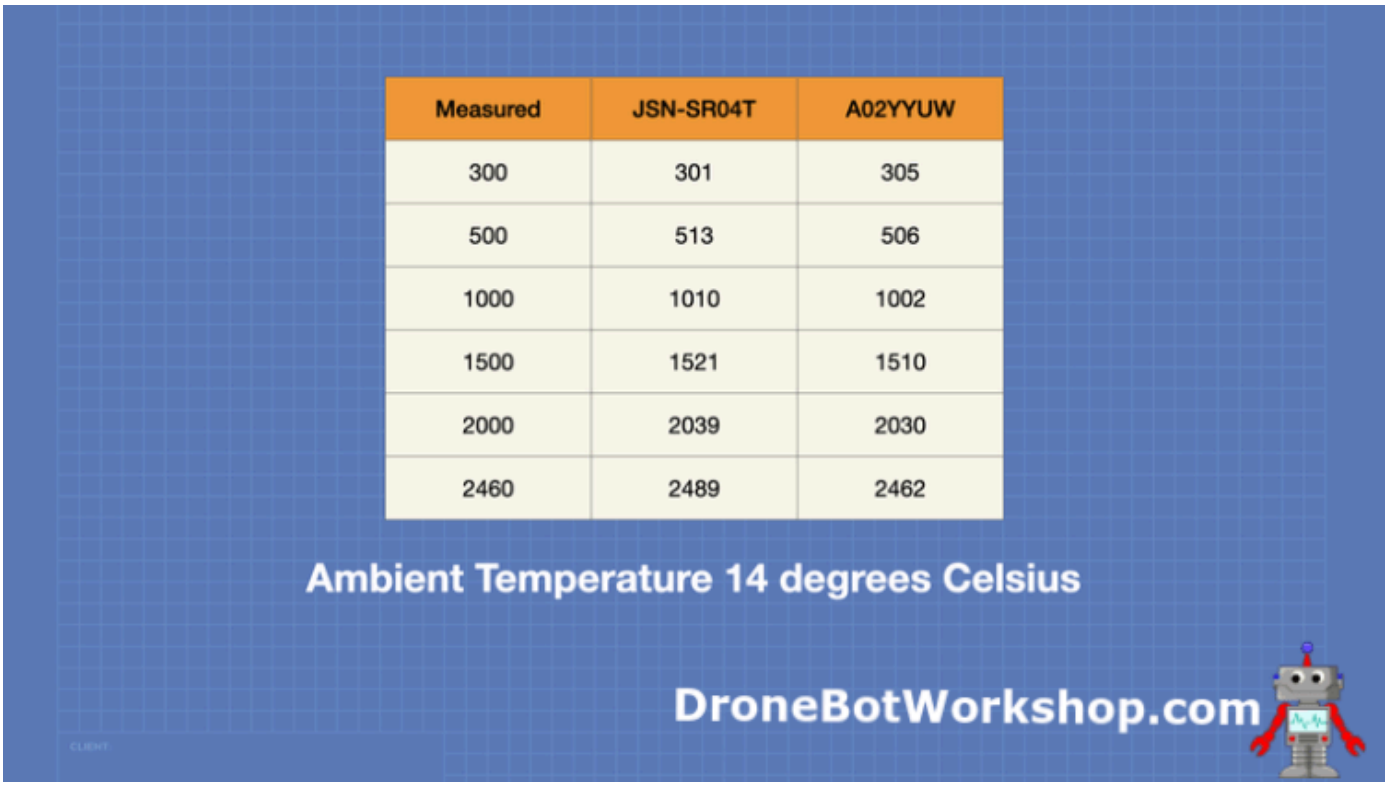
Outdoor Tests

As these sensors are waterproof I decided to give them a test outside.

My experiment was not in any way scientific, nor was it necessarily accurate. But it was sufficient to give me an idea as to how these sensors would perform outdoors. To simulate rain, I repeated the tests after wetting the surface of the sensors, being very careful not to water my Arduino or notebook computer!

The results were pretty good. I was especially pleased to see that the water had virtually no effect on the readings.

Here are my readings, again please remember that my crude testing setup probably affected the accuracy.



As you can see both sensors performed well at all distances. While they are not 100% accurate, they are more than sufficient for robotics, automotive, or intruder detection applications.

It's also worth noting that the temperature when I performed these measurements was 14 degrees Celsius, which is a bit on the cool side. As the speed of sound is affected by temperature, this probably affected my results.

Also, due to the small size of my yard, I was not able to determine the maximum range.

Either of these devices would work well in an outdoor application.

Underwater Tests

This last test was a bit of a joke, really, and I wasn't sure if I would get any results at all. And it turns out that I didn't!

I wanted to see if these sensors could give any readings while completely submerged in water. And as I don't own a swimming pool, the biggest body of water I had available was my bathtub!

Now, of course, I wasn't expecting any accurate results, as the speed of sound underwater is very different than it is in the air. But I was hoping to at least see a change in the readings as I moved the senors around the tub.

Unfortunately, the experiment was a failure, at least as far as getting reading goes. Each sensor just gave me a fixed reading, no matter how much I moved it.

One positive aspect of the test, however, was that as soon as the sensors were extracted from the water, they worked perfectly. So submerging them completely in the water had no effect upon them.

It definitely proved one thing – both of these sensors are indeed waterproof!

Conclusion

If you need an ultrasonic distance sensor that is waterproof, then either of these units would make an excellent choice. They both performed well, and they are rated for harsh environments.

They are, however, more expensive than the HC-SR04, so if you don't require a rugged waterproof sensor then the latter is still a more economical choice.

For a robotics project that I'm building (an outdoor "rover" style of robot) I've decided to use the A02YYUW sensor, as it is capable of resolving closer items. Plus, with all the electronics self-contained, it will be easier to work with. But the JSN-SRT04T would also be a fine choice as well.

Hopefully, this article and its associated video will help you determine which sensor is best for your outdoor application.

Parts List

Here are some components that you might need to complete the experiments in this article. Please note that some of these links may be affiliate links, and the DroneBot Workshop may receive a commission on your purchases. This does not increase the cost to you and is a method of supporting this ad-free website.

COMING SOON!

Resources

Code Samples – The Arduino code used in this article.

A02YYUW – Application notes from DF Robot

JSN-SR04T – Manufacturers spec sheet – please note this is for revision 2 of the unit, I used revision 3 in the article.