AZ-Delivery

# Welcome!

Thank you for purchasing our ***AZ-Delivery HC-SR04 Ultrasonic Sensor Module***. On the following pages, we will introduce you to how to use and set-up this handy device.

**Have fun!**

HC-SR04 Ultrasonic Sensor uses ultrasonic waves, to determine distance to the objects. The HC-SR04 module can do measurements without a contact in range from *20* milimeters to *4000* milimeters with just a *3* milimeters precision. Every HC-SR04 module has ultrasonic transmitter, receiver and electronic circuit.

Sensor consists of an ultrasonic transmitter and a receiver. Sensor transmits ultrasonic waves on a specific frequency. Ultrasonic wave travels through the air and hits an obstacle, then bounces back from the obstacle and gets detected by a receiver. The speed of the ultrasonic waves in the air is around *343m/s*, so the only thing to do is to measure the time interval, from transmitting to receiving. To calculate the distance, multiply speed of the ultrasonic wave in the air with half of the time interval. This time interval has to be divided by two, because ultrasonic wave travels to the obstacle and back.

## Specifications:

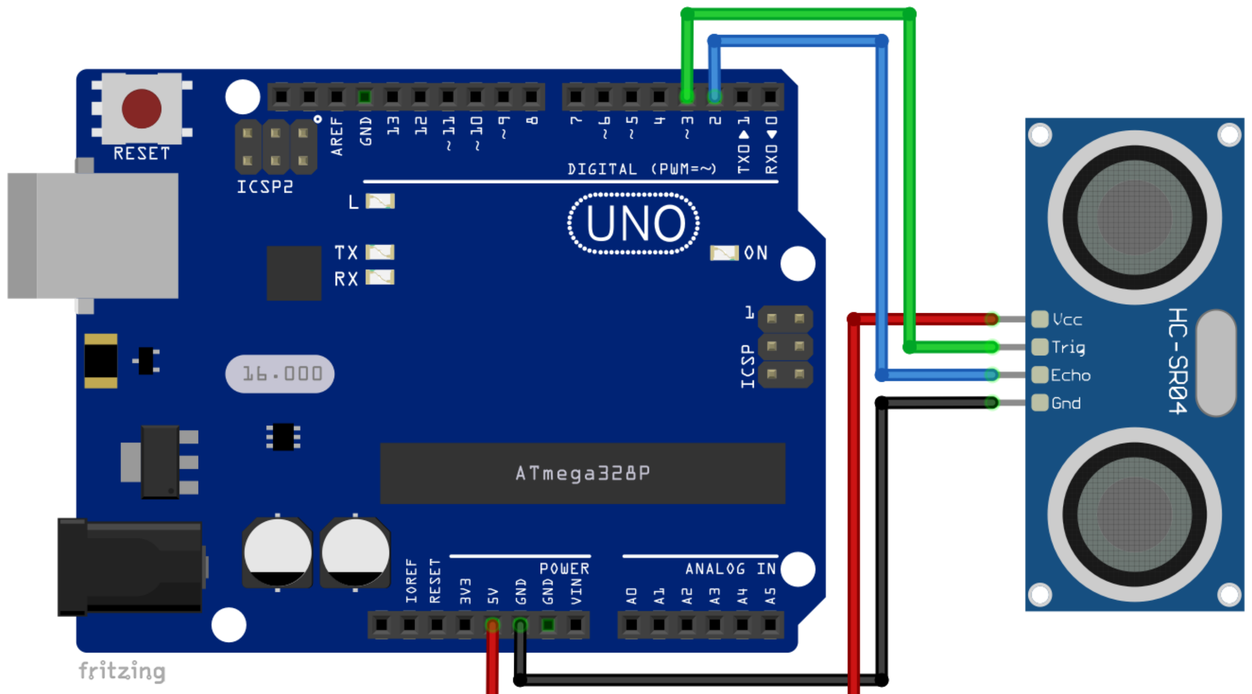
| | | |
|---|---|---|
| » | Power supply and logic voltage range: | 5V |
| » | Working DC current: | 15mA |
| » | Angle of measurement: | 15° |
| » | Distance range: | from 20mm to 4000mm |
| » | Practical measuring distance: | from 20mm to 800mm |

HC-SR04 sensor is used with both microcontoller and microprocessor platforms, such as Uno or Raspberry Pi. Power supply and logic voltage are *5V*. The working current is less than *15mA* and can be directly powered with *5V* pins. Raspberry Pi works on *3.3V*, so, in order to use it, we have to convert voltages from *5V* to *3.3V* with the device called *Logic Level Converter or Shifter*.

When we connect the module with microcontroller, we need two pins to do the measurement: triger pin as an input pin, and echo pin as an output pin. The trigger pin has to be set to *HIGH* state for *10us* and then set to *LOW* state. This transmits the ultrasonic wave, and the receiver waits for the wave to bounce back. When the wave returns, echo pin goes *HIGH*.

AZ-Delivery

## Connecting the module with Uno



| **HC-SR04 pin** | **>** | **Uno pin** | |
|---|---|---|---|
| VCC | > | 5V | **Red wire** |
| GND | > | GND | **Black wire** |
| ECHO | > | D2 | **Blue wire** |
| TRIG | > | D3 | **Green wire** |

**Sketch example:**

```
const int echoPin = 2;
const int trigPin = 3;
long duration;
int distance;
void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
}
void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = duration * 0.034 / 2;
  Serial.print("Distance: ");
  Serial.println(distance);
  delay(1000);
}
```

At the beginning of the sketch, two constants are created, called *`echoPin`* and *`trigPin`*, and set their values are set to *2* and *3*, respectively. These values represent digital pins of the Uno that are connected to echo pin and trig pin of the sensor. Then, two variables are created, one for measuring the time called *`duration`*, and the second for the calculated distance, called *`distance`*. In the `setup()` function, we set pin modes of used pins; Echo pin is defined as *INPUT* and the Trig pin as *OUTPUT*. At the end of `setup()` function, we start the serial communication with baud rate of 9600.

In the `loop()` function, first we set the state of *trigPin* to LOW, and wait for 2us. Then, the state of the variable `trigPin` goes *HIGH*, and algorithm waits for *10us*. After this, state of the variable `trigPin` goes to *LOW* state.

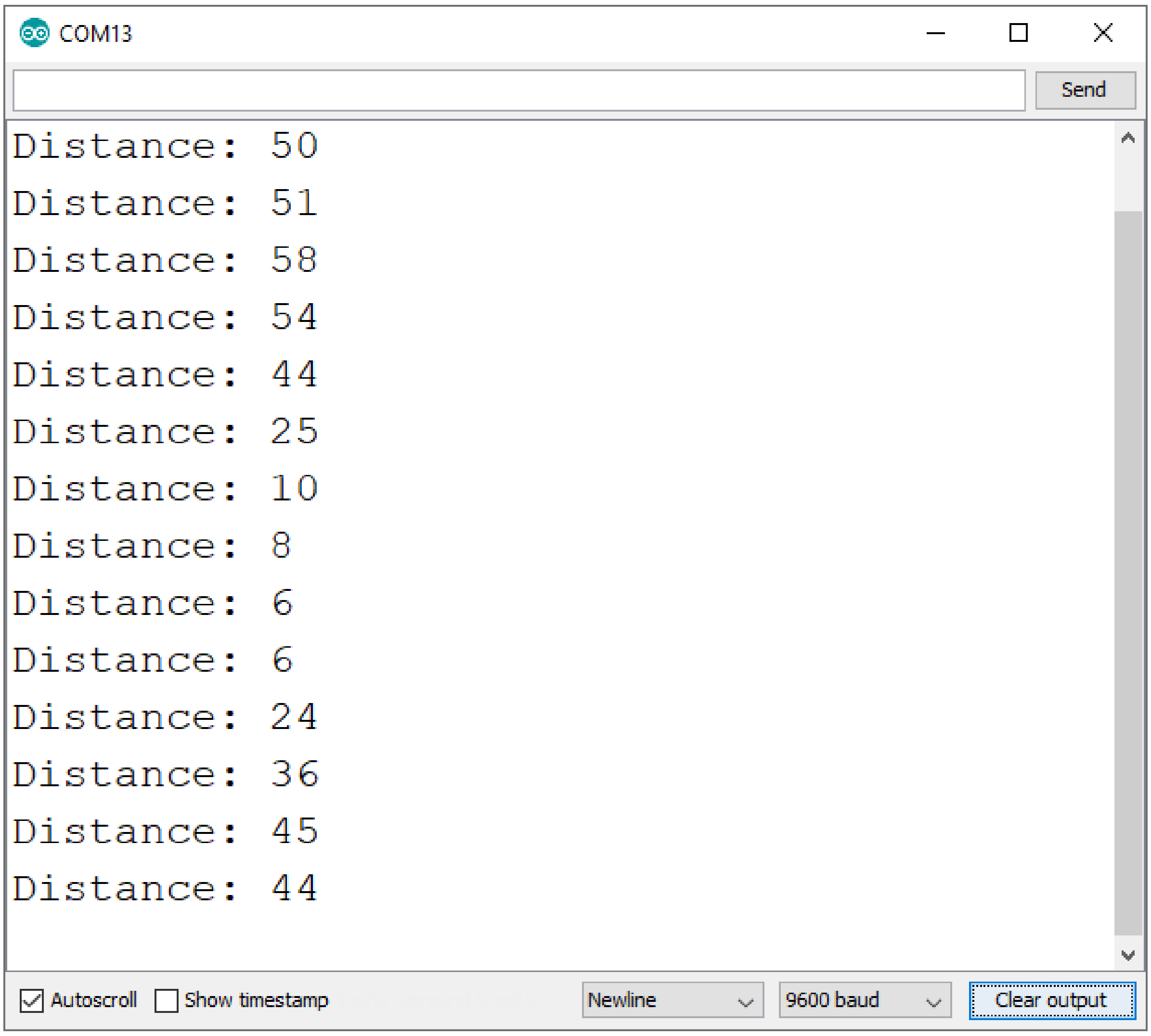
With the next line of code, we mesure time interval between transmission and detection of the ultrasonic wave, and store it in the `duaration` variable: `duration = pulseIn(echoPin, HIGH);`

Then to calculate the distance, we use next line of code:

```
distance = duration * 0.034 / 2;
```

At the end of `loop()` function, we output the distance data on the Serial Monitor. Also, at the end of `loop()` function, there is one delay of 1000 miliseconds. This is delay between two measurements.

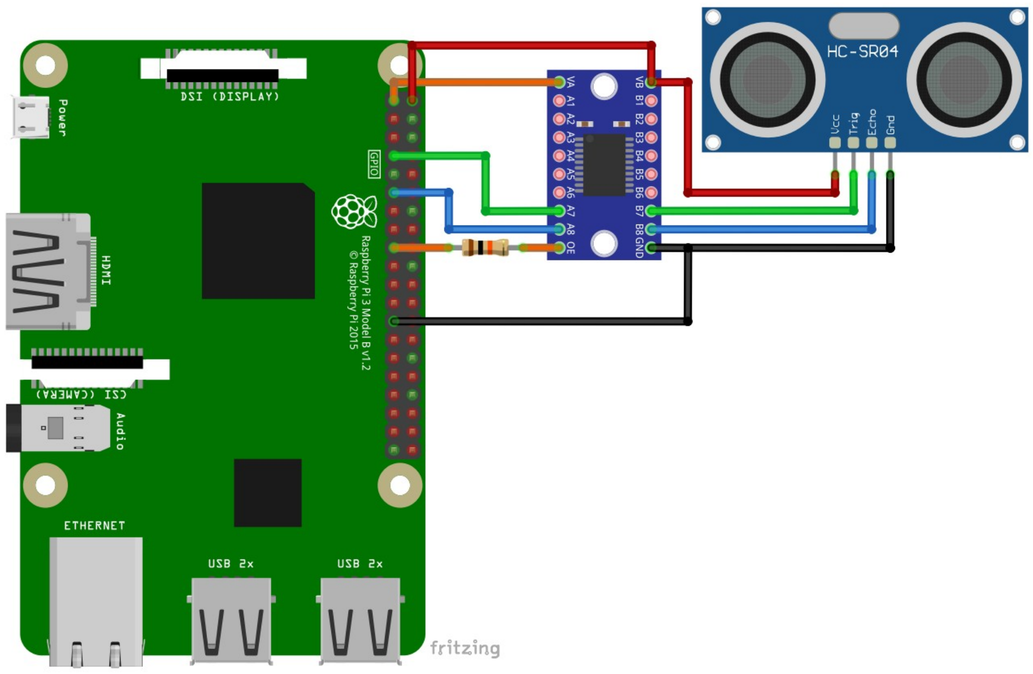
Upload the sketch to the Uno and start the Serial Monitor (*Tools* > *Serial Monitor*). The output should look like the output on the image below:



```
Distance:  50
Distance:  51
Distance:  58
Distance:  54
Distance:  44
Distance:  25
Distance:  10
Distance:  8
Distance:  6
Distance:  6
Distance:  24
Distance:  36
Distance:  45
Distance:  44
```

## Connecting the module with Raspberry Pi

Connect the sensor with the Raspberry Pi like on the connection diagram as shown below:



| **HC-SR04 pin** | **>** | **Raspberry Pi pin** | | |
|---|---|---|---|---|
| VCC | > | 5V | [pin 2] | **Red wire** |
| GND | > | GND | [pin 25] | **Black wire** |
| TRIG | > | GPIO4 | [pin 7] via LLC* | **Green wire** |
| ECHO | > | GPIO17 | [pin 11] via LLC* | **Blue wire** |

* LLC – Logic Level Converter

| **LLC pin** | **>** | **Raspberry Pi pin** | | |
|---|---|---|---|---|
| VA | > | 3V3 | [pin 1] | **Orange wire** |
| VB | > | 5V | [pin 2] | **Red wire** |
| GND | > | GND | [pin 25] | **Black wire** |
| OE | > | 3V3 | [pin 17] | **Orange wire** |

Connect between *OE* pin of the *LLC* and *3V3*, *10kΩ* pull up resistor.

The output of the HC-SR04 ultrasonic sensor are in the *5V* range, so the module output needs to be converted from *5V* to *3.3V* to avoid damaging the Raspberry Pi. This can be done with the module called Logic level converter or shifter. AZ-Delivery offers this kind of module called *"TXS0108E 8 channel logic level converter*". This module is bi-directional, it can convert voltages from *3.3V* to *5V,* and vice versa. It has 8 channels, which means that *8* different digital pins can be used for voltage conversions.

*VA* pin is used as a reference for lower level voltage (for example *3.3V* of the Raspberry Pi).

*VB* pin is used as a reference for higher level voltage (for example *5V* of the ultrasonic sensor).

*OE* pin or the Output Enable has to be connected to the *3.3V* via *10kΩ* pull up resistor.

Echo pin of the ultrasonic sensor is connected to the *B7* pin of the logic level converter, and the trig pin is connected to the *B6* pin of the logic level converter.

The *A7* pin of the logic level converter is connected to the *GPIO* pin *17*, and the *A6* pin of the logic level converter to the *GPIO* pin *4*.

**Python script:**

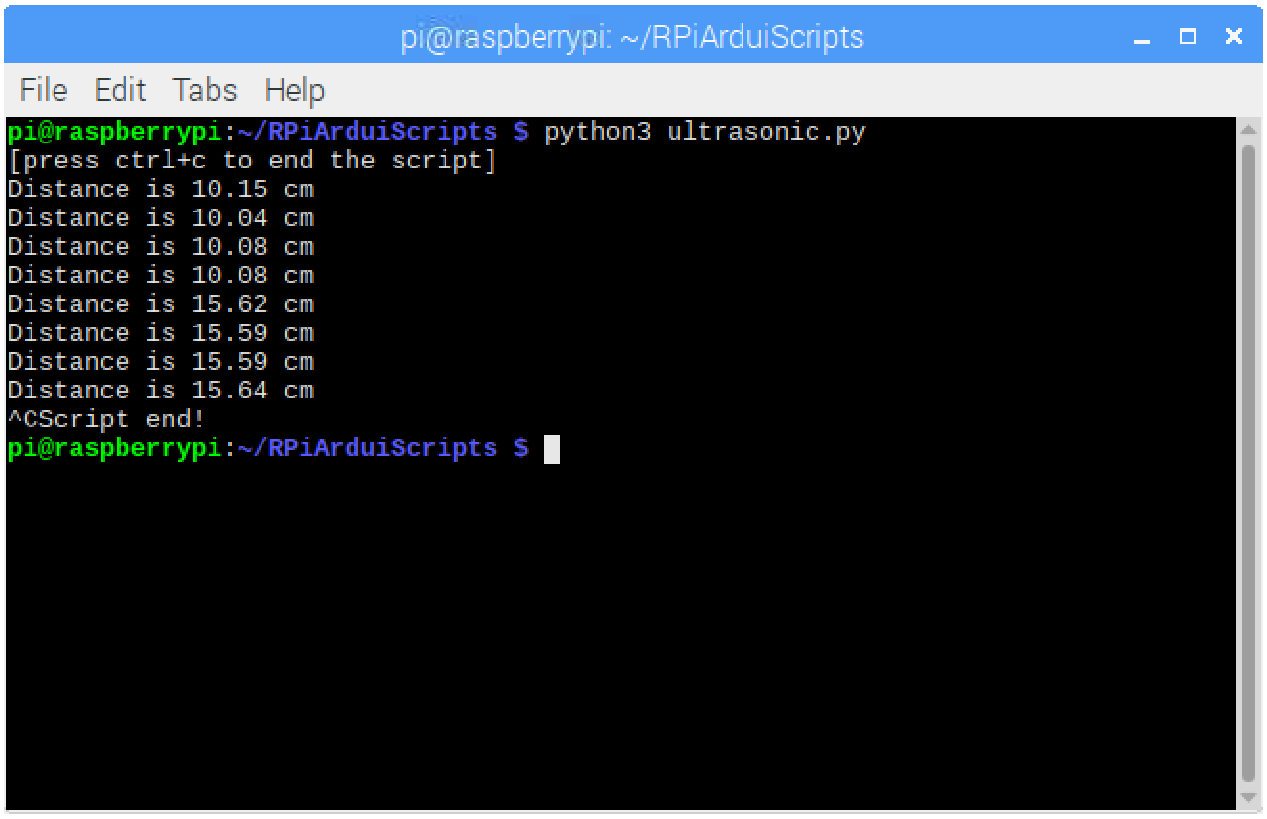
```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
TRIG = 17
ECHO = 4
GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)
GPIO.output(TRIG, False)
time.sleep(2)
print('[press ctrl+c to end the script]')
try: # Main program loop
      while True:
            GPIO.output(TRIG, True)
            time.sleep(0.00001)
            GPIO.output(TRIG, False)
            while GPIO.input(ECHO) == 0:
                  pulse_start = time.time()
            while GPIO.input(ECHO) == 1:
                  pulse_end = time.time()
            pulse_duration = pulse_end - pulse_start
            distance = pulse_duration * 17150
            distance = round(distance, 2)
            print('Distance is {} cm'.format(distance))
            time.sleep(2)

# Scavenging work after the end of the program
except KeyboardInterrupt:
      print('Script end!')

finally:
      GPIO.cleanup()
```

Save the script as “*ultrasonic.py*” and to run it open terminal and run this command: **`python3 ultrasonic.py`**

The output should look like the output on the image below:



```
pi@raspberrypi:~/RPiArduiScripts $ python3 ultrasonic.py
[press ctrl+c to end the script]
Distance is 10.15 cm
Distance is 10.04 cm
Distance is 10.08 cm
Distance is 10.08 cm
Distance is 15.62 cm
Distance is 15.59 cm
Distance is 15.59 cm
Distance is 15.64 cm
^CScript end!
pi@raspberrypi:~/RPiArduiScripts $
```

To stop the script press *CTRL + C*.

First, GPIO suite is introduced, which is basic GPIO control, and we turn off all warnings associated with GPIO pins. After that, we create and initialize two variables, called *TRIG* and *ECHO*. In these variables we store numbers *17* and *4* respectively. These numbers represent the GPIO pins that are used to connect sensor pins to.

After pin modes are set (*TRIG* pin as *OUTPUT*, and *ECHO* pin as *INPUT*), the state of the *TRIG* pin is set to *LOW*.

In the infinite loop block (`while True:`) ultrasonic wave has been transmitted, and the  the time of the transmission has been recorded. Then we wait for the ultrasonic wave to bounce back and get detected by ultrasonic receiver. This time has also been recorded.

Now, we transmit the ultrasonic pulse with this line of code `GPIO.output(TRIG, True)`,
and then listen for bounced ultrasonic wave.
“`While`” blocks are used to ensure that each signal timestamp is recorded in the correct order. The `time.time()` function will record latest timestamp for each timestamp. The recordings of these timestamps are done in two “`while`” blocks:

```
while GPIO.input(ECHO) == 0:
    pulse_start = time.time()

while GPIO.input(ECHO) == 1:
    pulse_end = time.time()
```

With this line of code we calculate the time interval between the transmission and detection of the ultrasonic wave:
`pulse_duration = pulse_end – pulse_start`

Because the ultrasonic wave travels from transmitter to the obstacle and back to the receiver, time interval in the variable `pulse_duaration` has to be divided by 2 when we calculate the distance to the obstacle.

To calculate the distance we use this line of code:
`distance = pulse_duration * 17150`

At the end of infinite loop block we round distance value to two decimal places, and output the data. Then we set the time interval between two measurements is to *2* seconds, in the `time.sleep(2)`.

To stop the script press the *CTRL* + *C*. This is called keyboard interrupt, and we wait for this in the except block: `except KeyboardInterrupt`

**You've done it!**

**Now you can use your module for various projects.**

Now is the time to learn and make the Projects on your own. You can do that with the help of many example scripts and other tutorials, which you can find on the internet.

**If you are looking for the high quality products for Arduino and Raspberry Pi, AZ-Delivery Vertriebs GmbH is the right company to get them from. You will be provided with numerous application examples, full installation guides, eBooks, libraries and assistance from our technical experts.**

https://az-delivery.de

Have Fun!

Impressum

https://az-delivery.de/pages/about-us