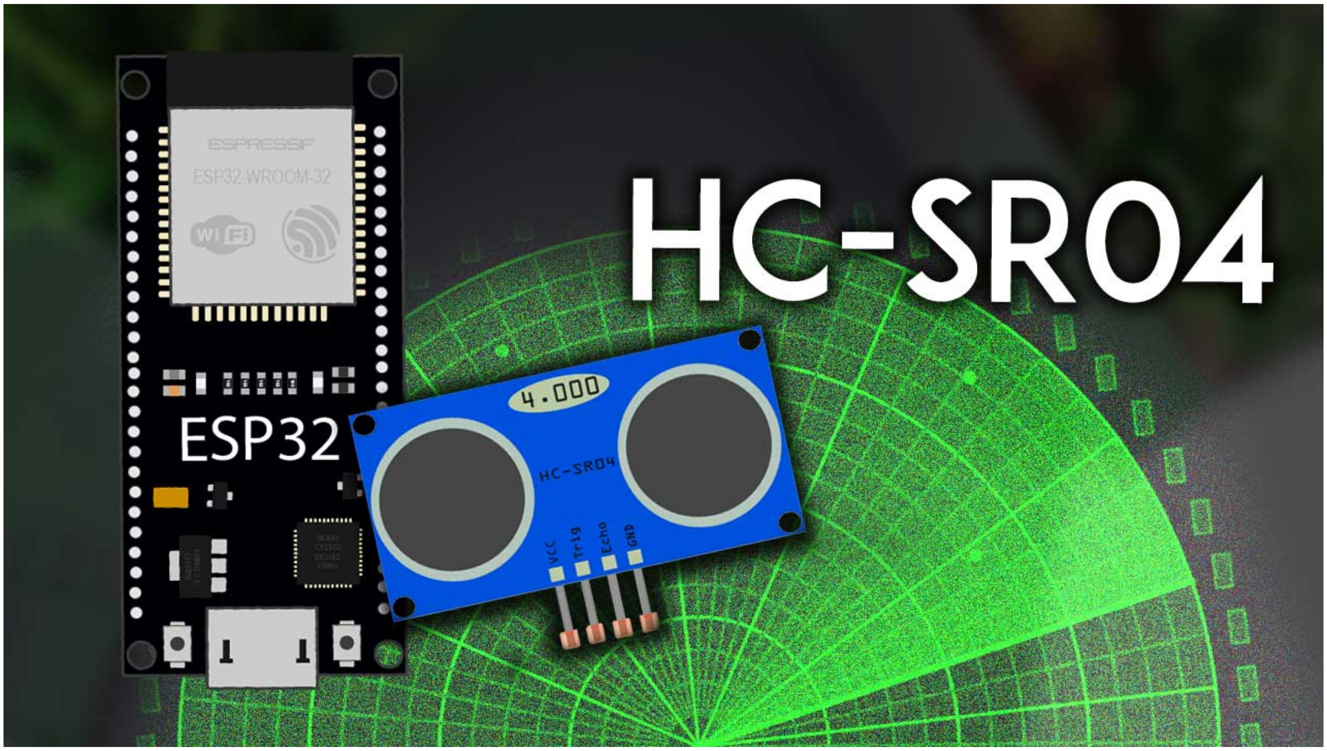
ESP32 with HC-SR04 Ultrasonic Sensor with Arduino IDE

This guide shows how to use the HC-SR04 Ultrasonic Sensor with the ESP32 board using the Arduino core. The ultrasonic sensor uses sonar to determine the distance to an object. We'll show you how to wire the sensor to the ESP32 and provide several example sketches to determine the distance to an object using the HC-SR04.



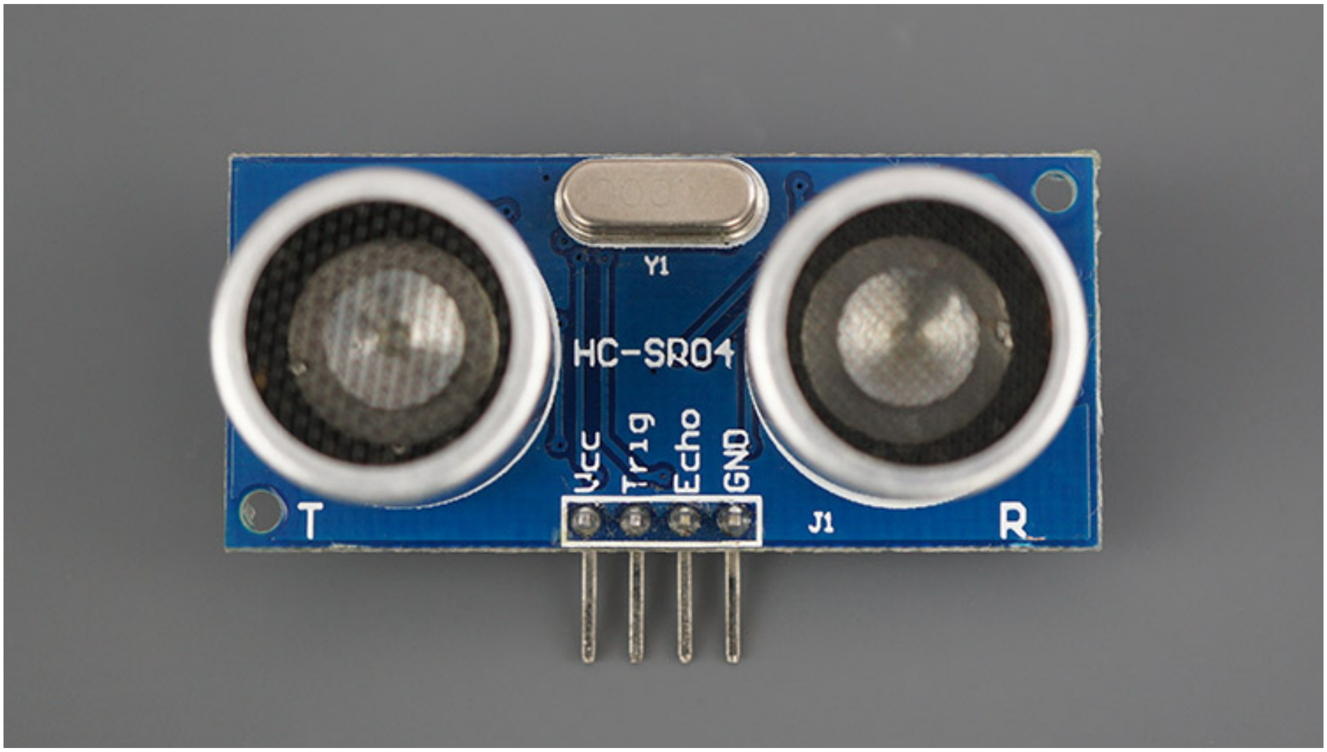
This tutorial covers the following topics:

- Ultrasonic Sensor HC-SR04 Pinout
- Wiring the Ultrasonic Sensor HC-SR04 to the ESP32
- Getting Distance to an Object Using the Ultrasonic Sensor HC-SR04 with the ESP32
- Displaying the Distance to an Object on a Display Using the ESP32 and HC-

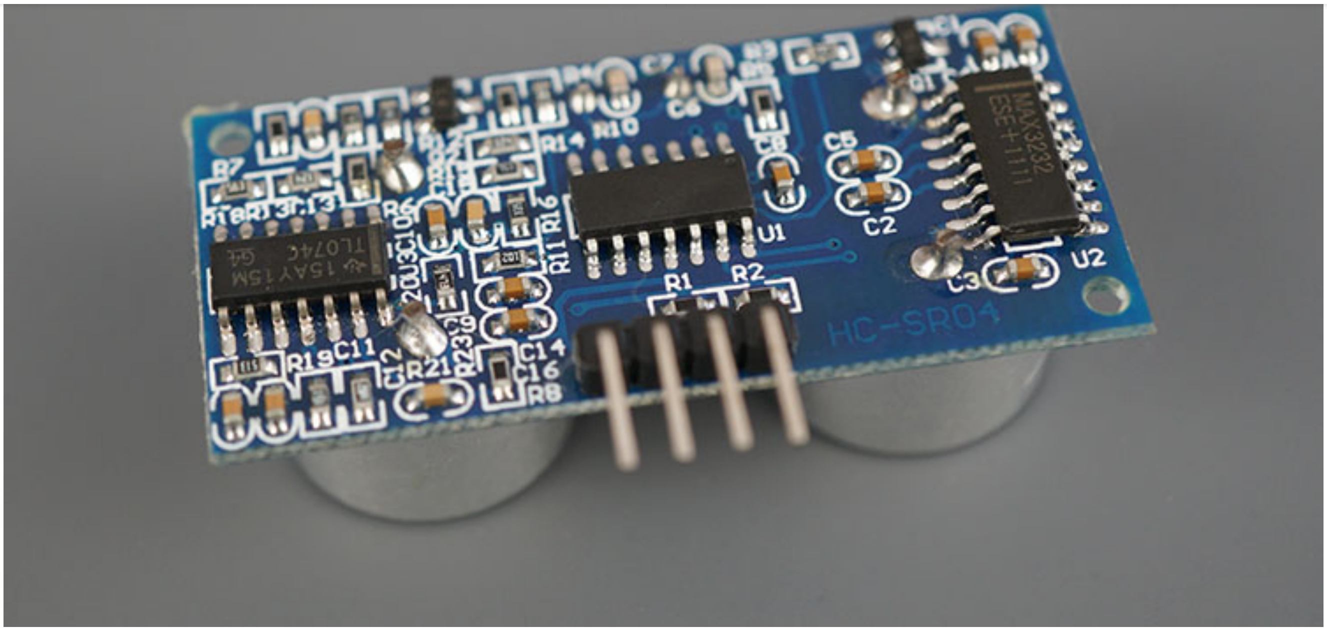
Introducing the HC-SR04 Ultrasonic Sensor

The HC-SR04 ultrasonic sensor uses sonar to determine the distance to an object. This sensor reads from 2cm to 400cm (0.8inch to 157inch) with an accuracy of 0.3cm (0.1inches), which is good for most hobbyist projects. In addition, this particular module comes with ultrasonic transmitter and receiver modules.

The following picture shows the HC-SR04 ultrasonic sensor.



The next picture shows the other side of the sensor.



Where to buy HC-SR04 Ultrasonic Sensor?

You can check the Ultrasonic Sensor HC-SR04 sensor on Maker Advisor to find the best price:

- HC-SR04 Ultrasonic Sensor

HC-SR04 Ultrasonic Sensor Technical Data

The following table shows the key features and specs of the HC-SR04 ultrasonic sensor. For more information, you should consult the sensor's datasheet.

Power Supply	5V DC
Working Current	15 mA
Working Frequency	40 kHz
Maximum Range	4 meters
Minimum Range	2 cm

Resolution	0.3 cm
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	TTL pulse proportional to the distance range
Dimensions	45mm x 20mm x 15mm

HC-SR04 Ultrasonic Sensor Pinout

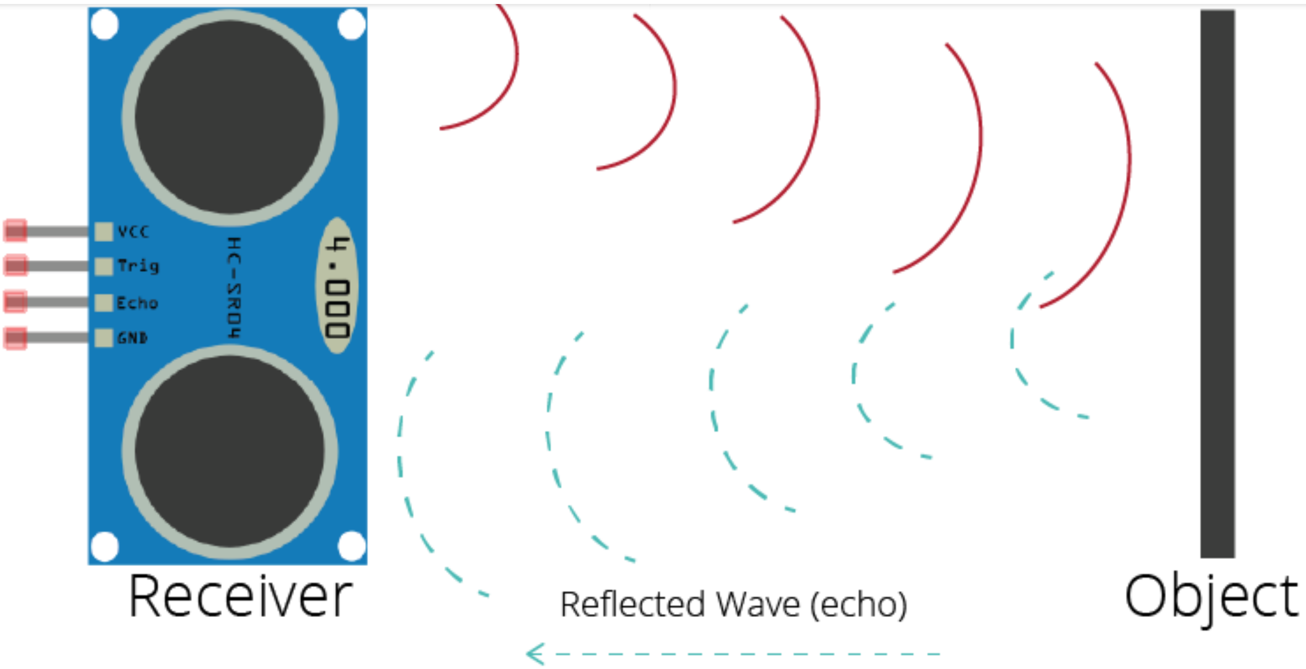
Here’s the pinout of the HC-SR04 Ultrasonic Sensor.

VCC	Powers the sensor (5V)
Trig	Trigger Input Pin
Echo	Echo Output Pin
GND	Common GND

How Does the HC-SR04 Ultrasonic Sensor Work?

The ultrasonic sensor uses sonar to determine the distance to an object. Here’s how it works:

1. The ultrasound transmitter (trig pin) emits a high-frequency sound (40 kHz).
2. The sound travels through the air. If it finds an object, it bounces back to the module.
3. The ultrasound receiver (echo pin) receives the reflected sound (echo).

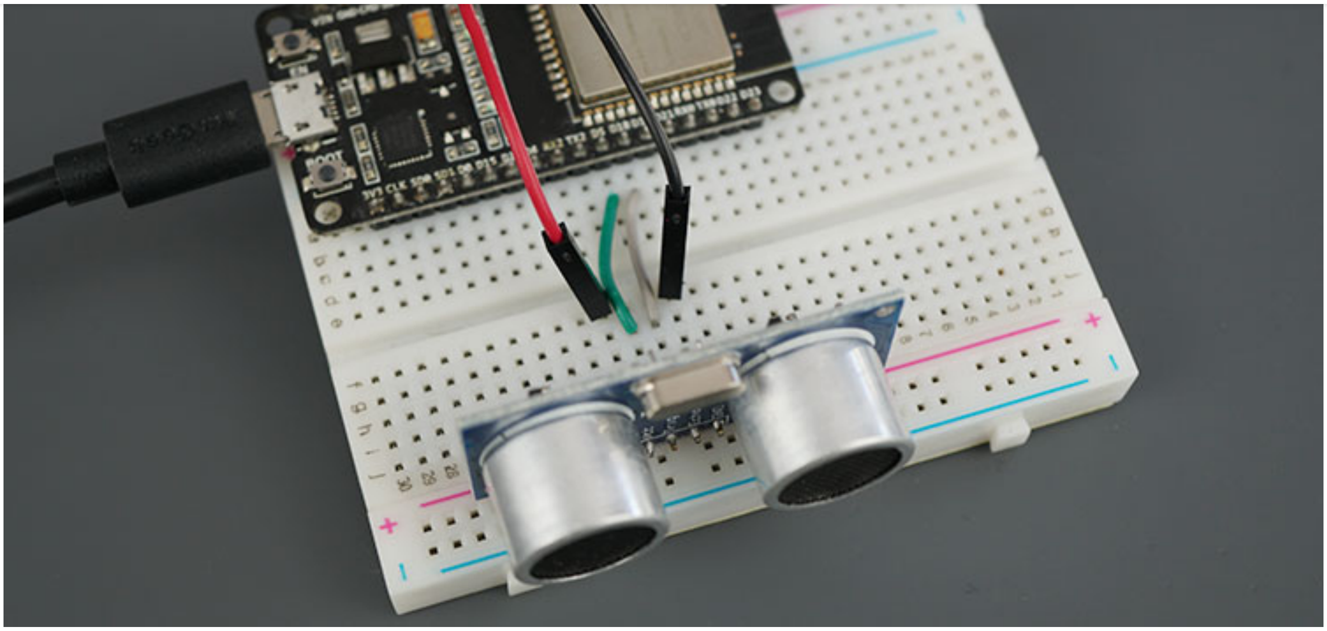


Taking into account the sound's velocity in the air and the travel time (time passed since the transmission and reception of the signal) we can calculate the distance to an object. Here's the formula:

```
distance to an object = ((speed of sound in the air)*time)/2
```

- speed of sound in the air at 20ºC (68ºF) = **343m/s**

Parts Required



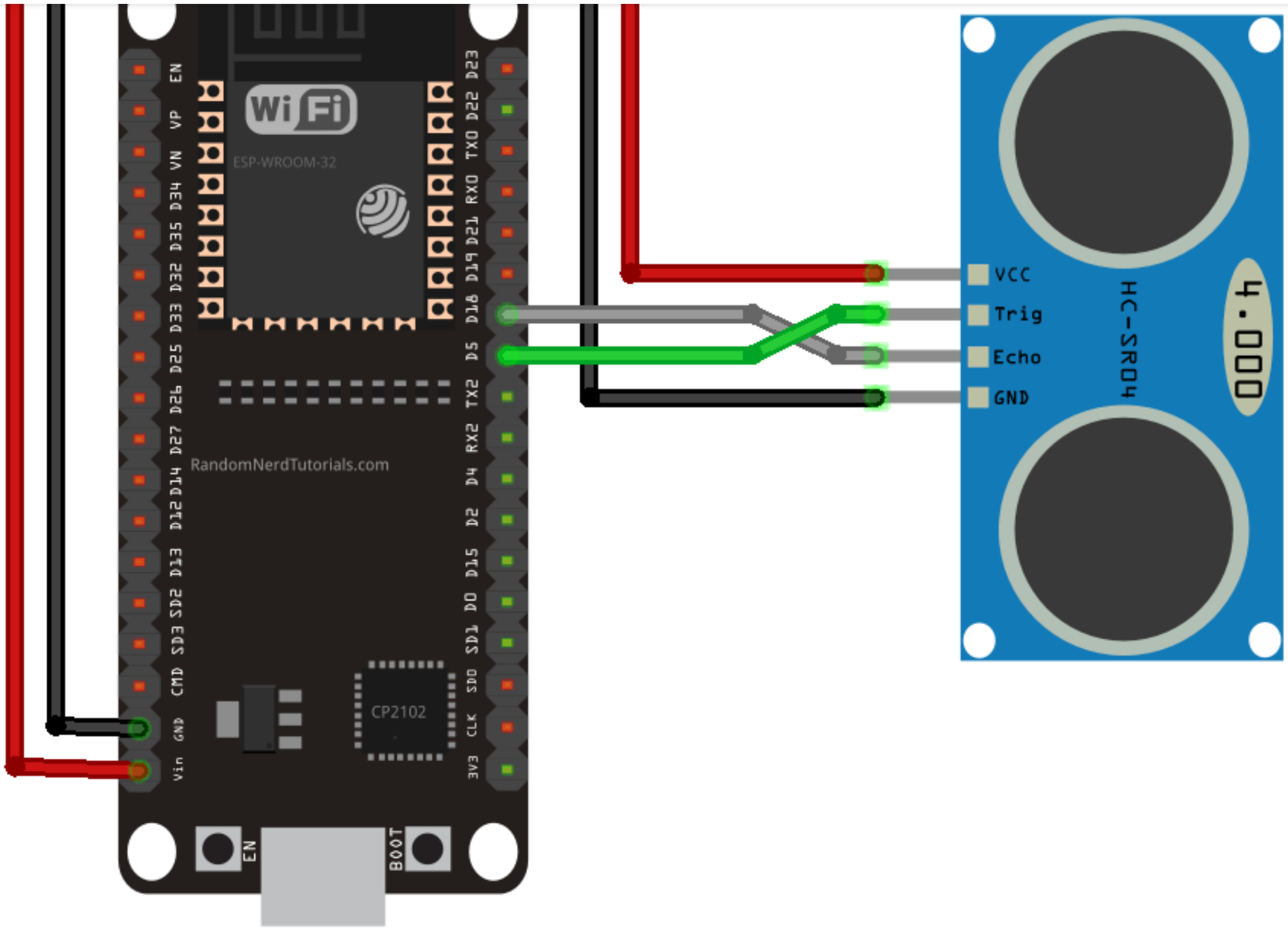
To complete this tutorial you need the following parts:

- HC-SR04 Ultrasonic Sensor
- ESP32 (read Best ESP32 development boards)
- Breadboard
- Jumper wires

You can use the preceding links or go directly to MakerAdvisor.com/tools to find all the parts for your projects at the best price!

Schematic – ESP32 with HC-SR04 Ultrasonic Sensor

Wire the HC-SR04 ultrasonic sensor to the ESP32 as shown in the following schematic diagram. We're connecting the Trig pin to GPIO 5 and the Echo pin to GPIO 18, but you can use any other suitable pins.



Ultrasonic Sensor	ESP32
VCC	VIN
Trig	GPIO 5
Echo	GPIO 18
GND	GND

Preparing Arduino IDE

We'll program the ESP32 board using Arduino IDE. So, make sure you have the ESP32 add-on installed. Follow the next tutorial:

If you want to use VS Code with the PlatformIO extension, follow the next tutorial instead to learn how to program the ESP32:

- Getting Started with VS Code and PlatformIO IDE for ESP32 and ESP8266 (Windows, Mac OS X, Linux Ubuntu)

Code – Getting Distance to an Object using the HC-SR04 Ultrasonic Sensor and ESP32

The following sketch is a simple example of how you can get the distance between the sensor and an object using the ESP32 board with the Arduino core.

```
/*********
  Rui Santos
  Complete project details at https://RandomNerdTutorials.co

  Permission is hereby granted, free of charge, to any perso
  of this software and associated documentation files.

  The above copyright notice and this permission notice shal
  copies or substantial portions of the Software.
*********/

const int trigPin = 5;
const int echoPin = 18;

//define sound speed in cm/uS
#define SOUND_SPEED 0.034
#define CM_TO_INCH 0.393701

long duration;
float distanceCm;
```

```
void setup() {
  Serial.begin(115200); // Starts the serial communication
  pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
  pinMode(echoPin, INPUT); // Sets the echoPin as an Input
}
```

View raw code

Upload the code to your board and it will work straight away. Continue reading if you want to learn how the code works or skip to the demonstration section.

How the Code Works

First, define the trigger and the echo pins.

```
const int trigPin = 5;
const int echoPin = 18;
```

In this example, we're using `GPIO 5` and `GPIO 18`. But you can use any other suitable GPIOs—read ESP32 Pinout Reference: Which GPIO pins should you use?

The `SOUND_SPEED` variable saves the velocity of sound in the air at 20ºC. We're using the value in cm/uS.

```
#define SOUND_SPEED 0.034
```

The `CM_TO_INCH` variable allows us to convert distance in centimeters to inches.

```
#define CM_TO_INCH 0.393701
```

```
long duration;
float distanceCm;
float distanceInch;
```

The `duration` variable saves the travel time of the ultrasonic waves (time elapsed since transmission and reception of the pulse wave). The `distanceCm` and `distanceInch`, as the names suggest, save the distance to an object in centimeters and inches.

setup()

In the `setup()`, initialize a serial communication at a baud rate of 115200 so that we can print the measurements on the Serial Monitor.

```
Serial.begin(115200); // Starts the serial communication
```

Define the trigger pin as an `OUTPUT`—the trigger pin emits the ultrasound. And define the echo pin as an `INPUT`—the echo pin receives the reflected wave and sends a signal to the ESP32 that is proportional to the travel time.

```
pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin, INPUT); // Sets the echoPin as an Input
```

loop()

In the `loop()`, the following lines produce a 10uS `HIGH` pulse on the trigger pin—this means the pin will emit an ultrasound. Note that before sending the pulse, we give a short `LOW` pulse to ensure you'll get a clean `HIGH` pulse.

```
// Clears the trigPin
// Sets the trigPin on HIGH state for 10 micro seconds
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
```

We use the `pulseIn()` function to get the sound wave travel time:

```
duration = pulseIn(echoPin, HIGH);
```

> The `pulseIn()` function reads a HIGH or a LOW pulse on a pin. It accepts as arguments the pin and the state of the pulse (either HIGH or LOW). It returns the length of the pulse in microseconds. The pulse length corresponds to the time it took to travel to the object plus the time traveled on the way back.

Then, we simply calculate the distance to an object taking into account the sound speed.

```
distanceCm = duration * SOUND_SPEED/2;
```

Convert the distance to inches:

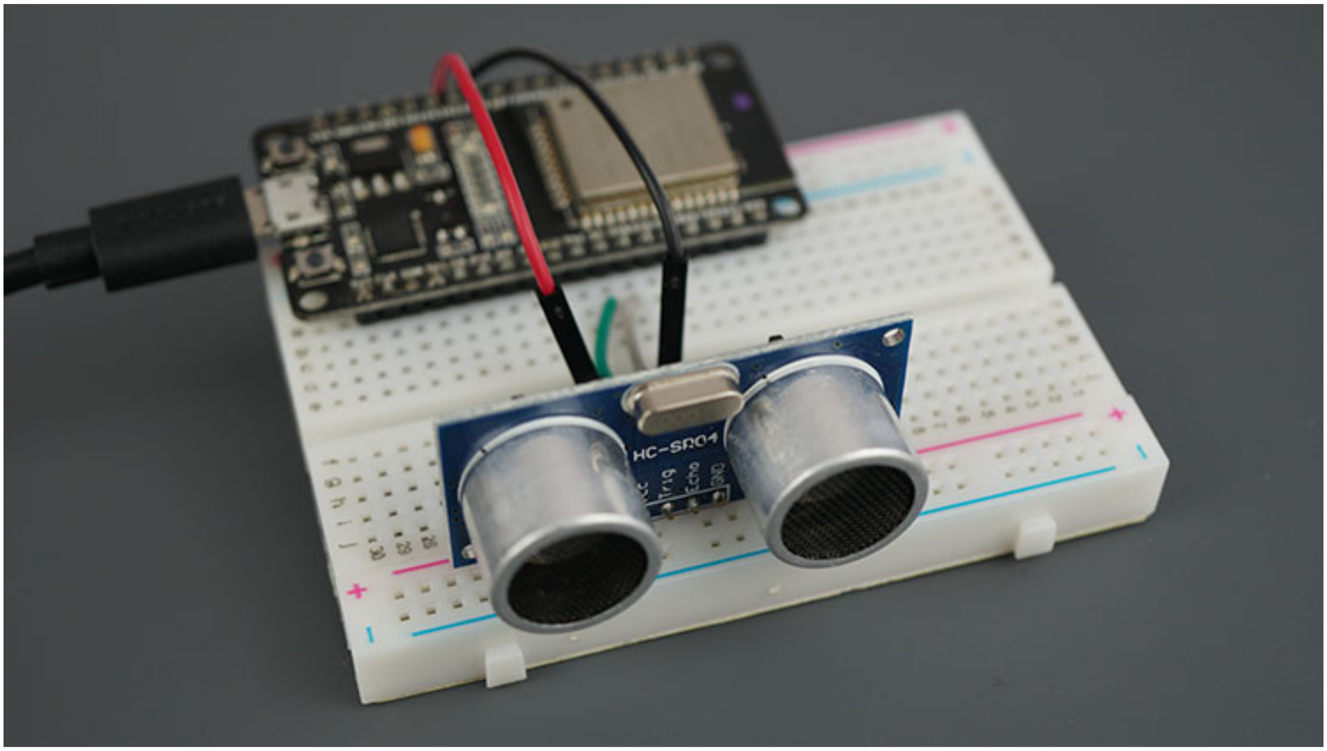
```
distanceInch = distanceCm * CM_TO_INCH;
```

And finally, print the results on the Serial Monitor.

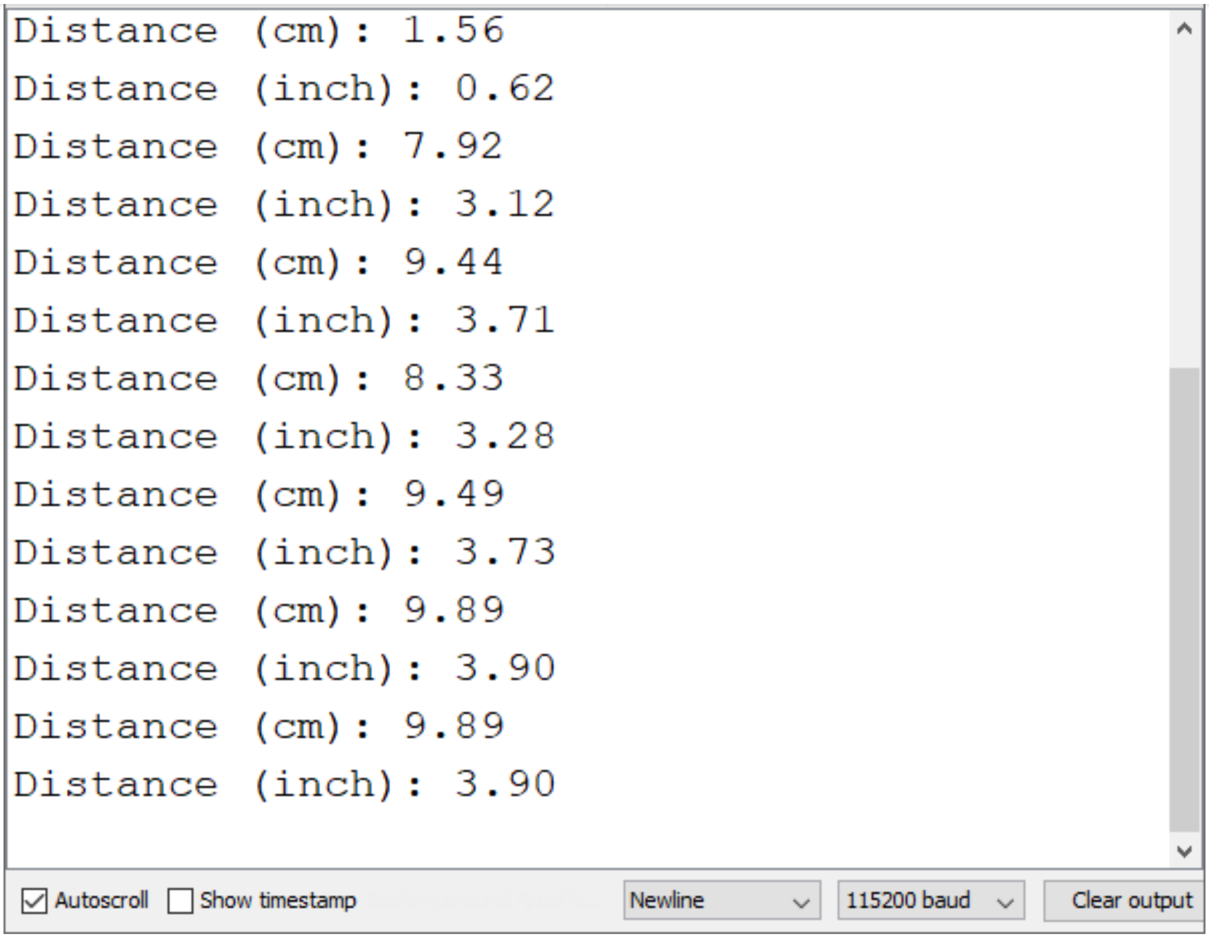
```
Serial.print("Distance (cm): ");
Serial.println(distanceCm);
Serial.print("Distance (inch): ");
```

Demonstration

Upload the code to your board. Don't forget to select the board you're using in **Tools** > **Boards**. Also, don't forget to select the right COM port in **Tools** > **Port**.



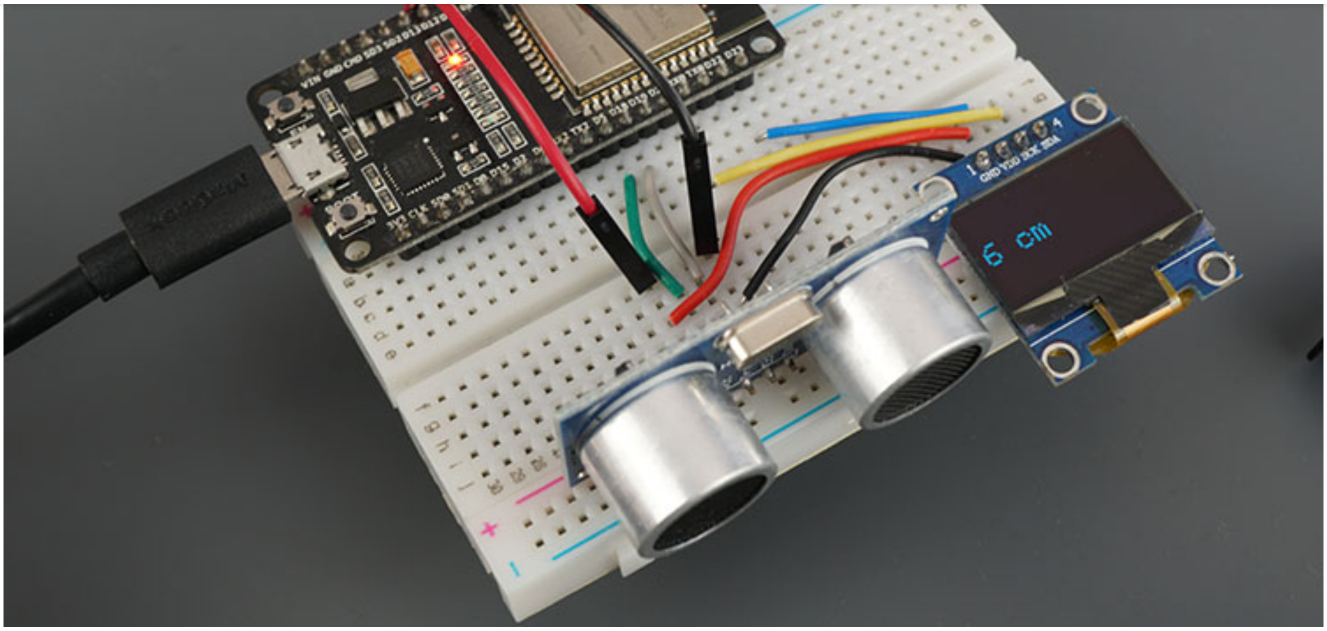
After uploading, open the Serial Monitor at a baud rate of 115200. Press the on-board RST button to restart the board and it will start printing the distance to the closest object on the Serial Monitor. Something as shown in the picture below.



```
Distance (cm): 1.56
Distance (inch): 0.62
Distance (cm): 7.92
Distance (inch): 3.12
Distance (cm): 9.44
Distance (inch): 3.71
Distance (cm): 8.33
Distance (inch): 3.28
Distance (cm): 9.49
Distance (inch): 3.73
Distance (cm): 9.89
Distance (inch): 3.90
Distance (cm): 9.89
Distance (inch): 3.90
```

☑ Autoscroll □ Show timestamp Newline 115200 baud

ESP32 with HC-SR04 and OLED Display



In this section, we'll show you a simple example with the ESP32 that displays the distance on an I2C OLED display.

To better understand how the project works, we recommend taking a look at our ESP32 tutorial with the I2C OLED display.

Parts Required

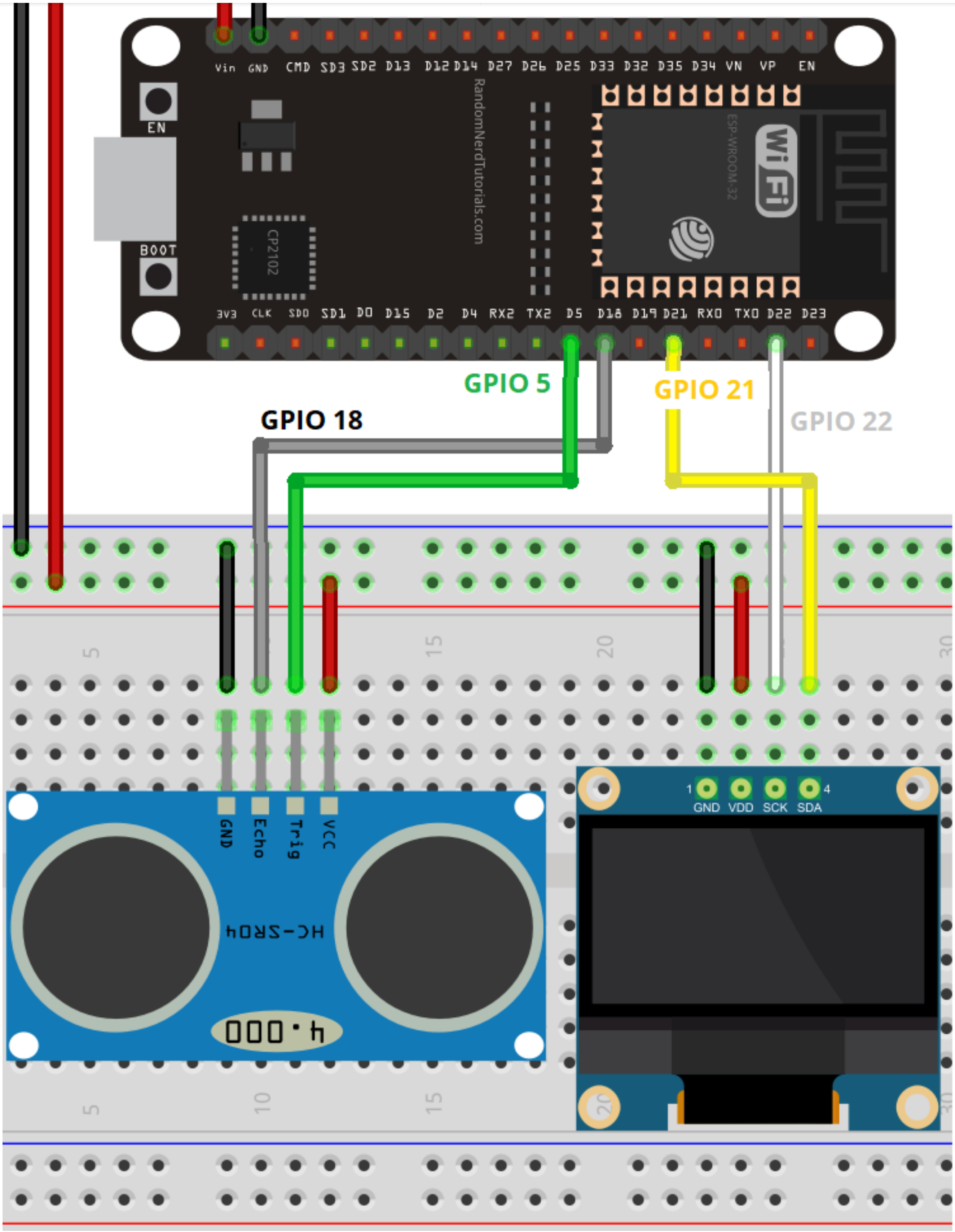
Here's a list with the parts required to complete this example:

- HC-SR04 Ultrasonic Sensor
- ESP32 (read Best ESP32 development boards)
- 0.96 inch I2C OLED Display SSD1306
- Breadboard
- Jumper wires

You can use the preceding links or go directly to MakerAdvisor.com/tools to find all the parts for your projects at the best price!

Schematic Diagram – ESP32 with HC-SR04 and OLED Display

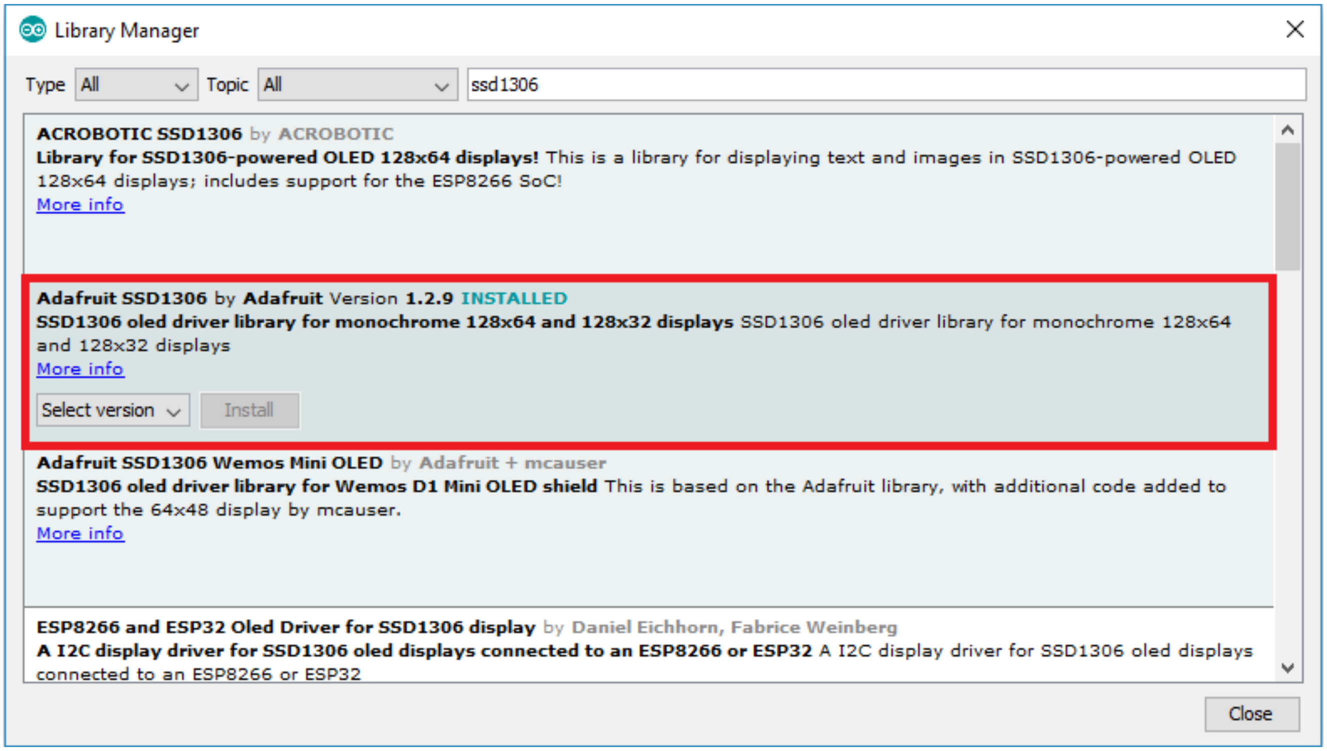
Wire all the parts as shown in the following schematic diagram.


Vin
GND
CMD
SD3
SD2
D13
D12
D14
D27
D26
D25
D33
D32
D35
D34
VN
VP
EN
EN
BOOT
CP2102
RandomNerdTutorials.com
ESP-WROOM-32
Wi Fi
3V3
CLK
SDO
SD1
D0
D15
D2
D4
RX2
TX2
D5
D18
D19
D21
RX0
TX0
D22
D23
GPIO 5
GPIO 18
GPIO 21
GPIO 22
GND
Echo
Trig
VCC
HC-SR04
4.000
1
GND
VDD
SCK
SDA
4
5
10
15
20
30

Code – ESP32 Display Distance (HC-SR04) on OLED Display

To use this example, make sure you have the Adafruit SSD1306 and Adafruit GFX libraries installed. You can install these libraries through the Arduino Library Manager.

Go to **Sketch** > **Library** > **Manage Libraries**, search for "**SSD1306,**" and install the SSD1306 library from Adafruit.



ssd1306

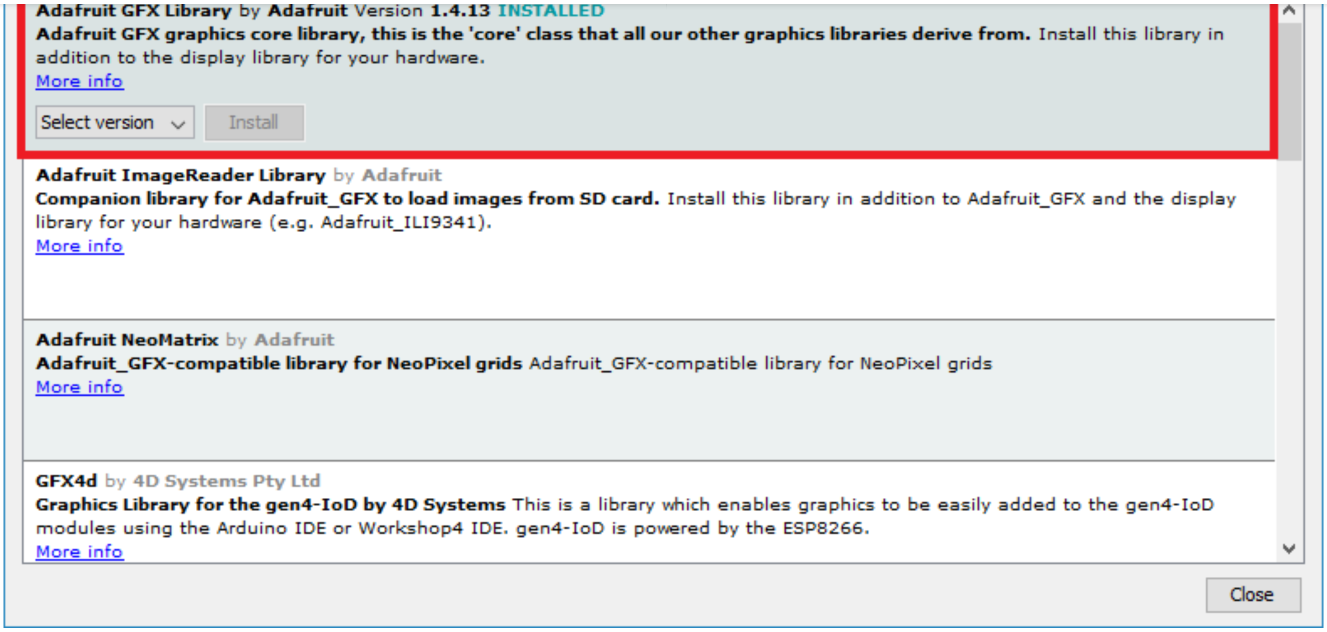
ACROBOTIC SSD1306 by ACROBOTIC
Library for SSD1306-powered OLED 128x64 displays! This is a library for displaying text and images in SSD1306-powered OLED 128x64 displays; includes support for the ESP8266 SoC!
More info

Adafruit SSD1306 by **Adafruit** Version **1.2.9** INSTALLED
SSD1306 oled driver library for monochrome 128x64 and 128x32 displays SSD1306 oled driver library for monochrome 128x64 and 128x32 displays
More info

Adafruit SSD1306 Wemos Mini OLED by Adafruit + mcauser
SSD1306 oled driver library for Wemos D1 Mini OLED shield This is based on the Adafruit library, with additional code added to support the 64x48 display by mcauser.
More info

ESP8266 and ESP32 Oled Driver for SSD1306 display by Daniel Eichhorn, Fabrice Weinberg
A I2C display driver for SSD1306 oled displays connected to an ESP8266 or ESP32 A I2C display driver for SSD1306 oled displays connected to an ESP8266 or ESP32

After installing the SSD1306 library from Adafruit, type "**GFX**" in the search box and install the library.



Adafruit GFX Library by Adafruit Version 1.4.13 INSTALLED
Adafruit GFX graphics core library, this is the 'core' class that all our other graphics libraries derive from. Install this library in addition to the display library for your hardware.
More info

Adafruit ImageReader Library by Adafruit
Companion library for Adafruit_GFX to load images from SD card. Install this library in addition to Adafruit_GFX and the display library for your hardware (e.g. Adafruit_ILI9341).
More info

Adafruit NeoMatrix by Adafruit
Adafruit_GFX-compatible library for NeoPixel grids Adafruit_GFX-compatible library for NeoPixel grids
More info

GFX4d by 4D Systems Pty Ltd
Graphics Library for the gen4-IoD by 4D Systems This is a library which enables graphics to be easily added to the gen4-IoD modules using the Arduino IDE or Workshop4 IDE. gen4-IoD is powered by the ESP8266.
More info

After installing the libraries, restart your Arduino IDE.

Then, simply copy the following code to your Arduino IDE and upload the code to the board.

```
/*********
  Rui Santos
  Complete project details at https://RandomNerdTutorials.cc

  Permission is hereby granted, free of charge, to any perso
  of this software and associated documentation files.

  The above copyright notice and this permission notice shal
  copies or substantial portions of the Software.
*********/

#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

```
// Declaration for an SSD1306 display connected to I2C (SDA,
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,

const int trigPin = 5;
const int echoPin = 18;

//define sound speed in cm/uS
#define SOUND_SPEED 0.034
#define CM_TO_INCH 0.393701
```

View raw code

How the Code Works

Start by including the required libraries for the OLED display:

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
```

Define the width and height of the OLED display. We're using a 128×64 OLED display:

```
#define SCREEN_WIDTH 128 // OLED display width, in pixels
#define SCREEN_HEIGHT 64 // OLED display height, in pixels
```

Create an `Adafruit_SSD1306` object called `display` to handle the OLED display.

Define the pins that the HC-SR04 sensor is connected to.

```
const int trigPin = 5;
const int echoPin = 18;
```

Create variables to save the distance and the duration between the transmission and reception of the sound waves.

```
long duration;
int distanceCm;
int distanceInch;
```

setup()

In the `setup()`, initialize a serial communication at a baud rate of 115200 so that we can print the results on the Serial Monitor.

```
Serial.begin(115200);
```

Define the trigger pin as an `OUTPUT` and the echo pin as an `INPUT`.

```
pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
pinMode(echoPin, INPUT); // Sets the echoPin as an Input
```

Initialize the OLED display:

```
if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
  }
```

Set the font size and color for the display.

```
  display.setTextSize(2);
  display.setTextColor(WHITE);
```

loop()

In the `loop()` is where we'll get the distance and display it on the OLED.

Get the distance (we've already explained in the previous section how to calculate the distance).

```
  // Clears the trigPin
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  // Sets the trigPin on HIGH state for 10 micro seconds
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in
  duration = pulseIn(echoPin, HIGH);

  // Calculate the distance
  distanceCm = duration * SOUND_SPEED/2;

  // Convert to inches
  distanceInch = distanceCm * CM_TO_INCH;
  // Prints the distance on the Serial Monitor
  Serial.print("Distance (cm): ");
  Serial.println(distanceCm);
  Serial.print("Distance (inch): ");
  Serial.println(distanceInch);
```

Clear the display in each `loop()` to write new readings.

```
  display.clearDisplay();
```

Set the display cursor to (0, 25).

```
  display.setCursor(0, 25);
```

The following lines print the distance in centimeters in the OLED display.

```
  // Display static text
  display.print(distanceCm);
  display.print(" cm");
```

Comment the previous lines and uncomment the following lines if you want to display the readings in inches.

```
  /* Display distance in inches
  display.print(distanceInch);
  display.print(" in");*/
```

```
display.display();
```

The distance is updated every 500 milliseconds.

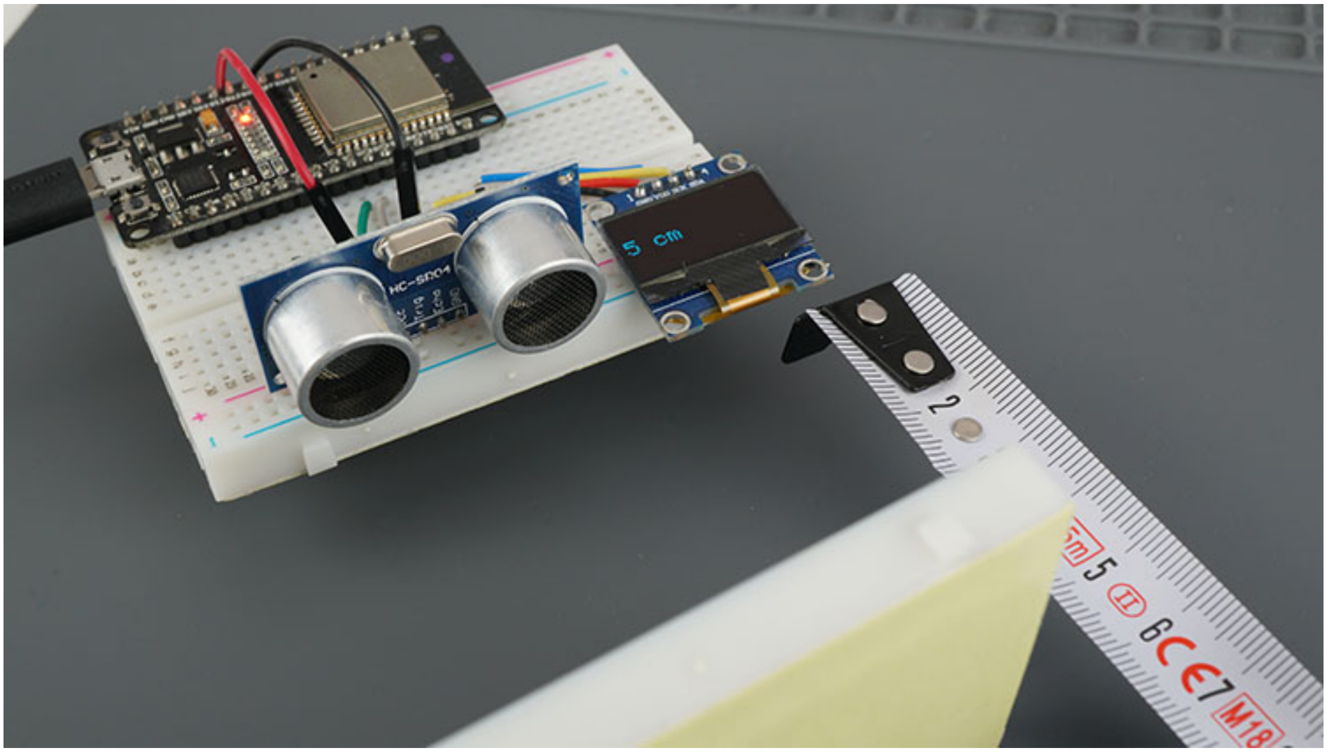
```
delay(500);
```

Demonstration

Upload the code to your ESP32 board. Go to **Tools** > **Board** and select the ESP32 board you're using. Go to **Tools** > **Port** and select the port your board is connected to. Then, click the upload button.

Open the Serial Monitor at a baud rate of 115200, press the on-board RST button. The sensor measurements will be displayed both on the Serial Monitor and the OLED display.

Approximate an object to the sensor and see the values changing.



00:17

Wrapping Up

The HC-SR04 Ultrasonic Sensor allows us to determine the distance to an object. In this tutorial you've learned how to use the HC-SR04 with the ESP32. We have tutorials for other popular sensors that you may like:

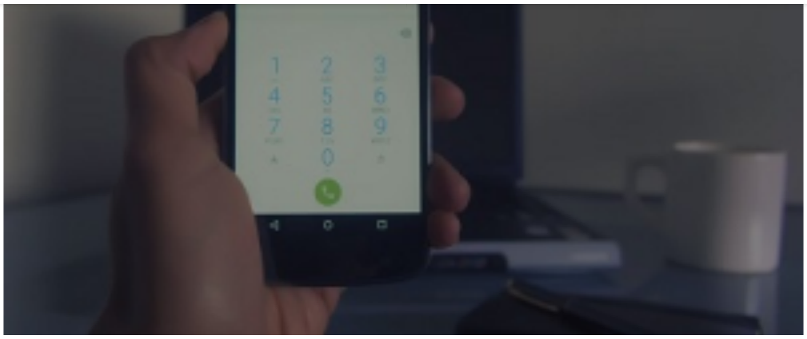
- ESP32 with **DHT11/DHT22 Temperature and Humidity Sensor** using Arduino IDE
- ESP32 with **BME280** using Arduino IDE (**Pressure, Temperature, Humidity**)
- ESP32 with **BME680 Environmental Sensor** using Arduino IDE (Gas, Pressure, Humidity, Temperature)
- ESP32 **DS18B20 Temperature Sensor** with Arduino IDE (Single, Multiple, Web Server)
- ESP32 with **BMP180 Barometric Sensor** (Temperature and Pressure)

Learn more about the ESP32 with our resources:

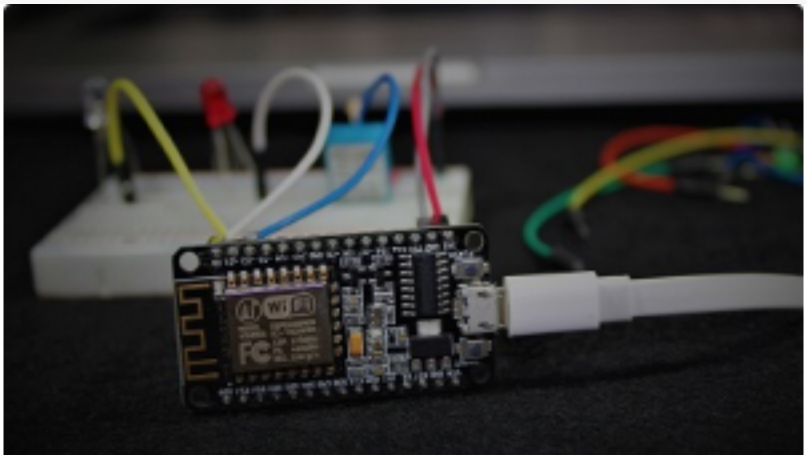
- More ESP32 Projects and Tutorials…

Thanks for reading.

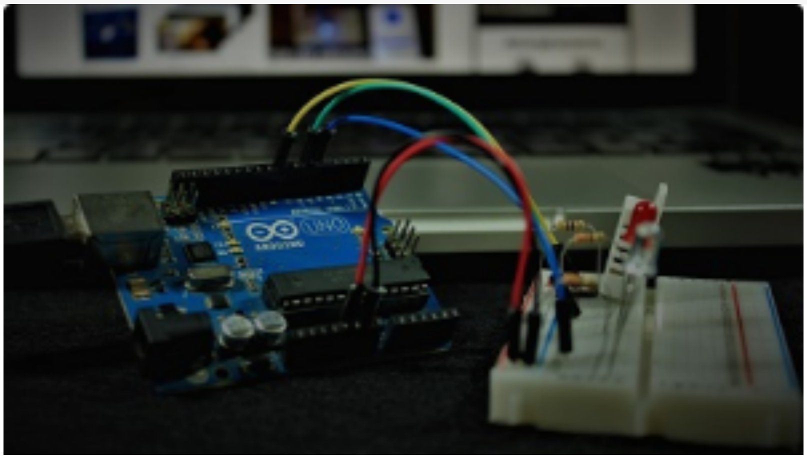
Recommended Resources



Build a Home Automation System from Scratch » With Raspberry Pi, ESP8266, Arduino, and Node-RED.



Home Automation using ESP8266 eBook and video course » Build IoT and home automation projects.



Arduino Step-by-Step Projects » Build 25 Arduino projects with our course, even with no prior experience!

What to Read Next...

Firebase: Control ESP8266 NodeMCU GPIOs from Anywhere

MicroPython: Program ESP32/ESP8266 using VS Code and Pymakr

ESP32 Web Server – Arduino IDE

39 thoughts on "ESP32 with HC-SR04 Ultrasonic Sensor with Arduino IDE"

Steve

July 20, 2021 at 7:16 pm

Excellent write-up; I hadn't realized how simple it is to get one of these up and running! I do have a question: you're using 5V to power the sensor, but the folks at Sparkfun say that the ESP32 uses 3.3V logic, and that there

(source: https://learn.sparkfun.com/tutorials/esp32-thing-hookup-guide/hardware-overview)

I saw a comment on a similar blog post that suggested a better choice of sensor would be the RCWL-1601, which is drop-in identical to the HC-SR04. A quick inquiry to Mr. Google suggests that the price and availability are about the same.

Reply

Dave

July 21, 2021 at 6:14 pm

I power the ultrasonic with 5v for more stable operation but connect the echo pin to the ESP via a level shifter as this is an output pin from the ultrasonic module and does output 5v which is too high for the 3v3 ESP module.
A 1k0 and 2k2 resistor in series works for me.

Reply

Andre Basel

July 20, 2021 at 8:13 pm

The RCWL-1601 is very cool. It’s a microwave sensor using doppler and so penetrates certain obstructions.

Steve

July 20, 2021 at 8:25 pm

@Andre, I think you're confusing the RCWL-1601, which is ultrasonic, exactly like the HC-SR04, with the RCWL-0516 which is in fact microwave/ doppler-based like you said. But it's a different beast entirely.

Compare:
RCWL-1601: https://www.adafruit.com/product/4007
RCWL-0516: https://dronebotworkshop.com/rcwl-0516-experiments/

Reply

Andre Basel

July 20, 2021 at 11:27 pm

@Steve, lol yeah you are right. Sorry all.

Except for the fact that the RCWL-0516 is a great little sensor, please ignore what I said 😉

Reply

Steve

July 21, 2021 at 12:34 am

seeing them on Aliexpress for about $0.75 each, shipped (to the US). Maybe this would be a good topic for the next Random Nerd tutorial…??? 😀

Reply

Andre Basel

July 21, 2021 at 1:15 am

I used it here for an IoT demo: https://www.instructables.com/IoT-Demo/ (the code used has a couple of bugs but is a good starting place I think).

Works like a treat. Going to use one for a letterbox notifier as it is nice and flat.

Reply

Ralph McCleery

July 20, 2021 at 10:13 pm

A subject near and dear to me so I thought I'd share some findings I have made in my several months of trial, error and testing with ESP devices and the HC-SR04 (both the standard and the Waterproof versions.)

#1 Your sketch for distance is a common method that was originally employed with AVR devices but I found that when used in conjunction with caused it as when I changed to an Arduino library it no longer crashed. The library I've had most success with to date on any ESP device is https://github.com/gamegine/HCSR04-ultrasonic-sensor-lib/

#2 You can in fact power the HC-SR04 with the 3.3v but you will need to increase the trigger time to 20us. I found anything less and you get too may false readings. My current test rig is an ESP12F on a BoB as it's intended to be a battery powered project.

#3 As distance increases you'll find reflections causes havoc on your readings if your sensor is near other hard surfaces. I've improved the reliability greatly by surrounding the sensor with acoustic (sponge) foam.

#4 To look at trying to improve the repeatability and accuracy for my project I have setup a sketch that has a smoothing array[10] and a single distance read and I have been comparing them for a month or two now. I round the values to MM and now at worst I get a variation of +/- 1mm between the two values.

#5 The next step to improve accuracy is to make the speed of sound constant a variable as there will be situation of large temperature changes when using the HC-SR04 outdoors.

#6 If intending to run the HC-SR04 as a battery project and employing deep sleep mode you need a minimum of 250ms between power on to the HC-SR04 and taking a reading. Any less and your reading will fail.
Cheers
Ralph

Reply

July 22, 2021 at 10:39 am

Thanks for the tips!

Reply

Riley
April 14, 2023 at 12:49 pm

Hello! I have an error in my code that says:

```
Le croquis utilise 273309 octets (20%) de l’espace de stockage de programmes. Le maximum est de 1310720 octets.
Les variables globales utilisent 22808 octets (6%) de mémoire dynamique, ce qui laisse 304872 octets pour les variables locales. Le maximum est de 327680 octets.
esptool.py v4.5.1
Serial port COM19
Connecting….
Chip is ESP32-D0WDQ6 (revision v1.0)
Features: WiFi, BT, Dual Core, 240MHz, VRef calibration in efuse, Coding Scheme None
Crystal is 40MHz
MAC: 08:3a:f2:7d:5f:d4
Uploading stub…
Running stub…
Stub running…
WARNING: Failed to communicate with the flash chip, read/write operations will fail. Try checking the chip connections or removing any
```

```
Flash will be erased from 0x00001000 to 0x00005fff…
Flash will be erased from 0x00008000 to 0x00008fff…
Flash will be erased from 0x0000e000 to 0x0000ffff…
Flash will be erased from 0x00010000 to 0x00052fff…
Compressed 17520 bytes to 12170…

A fatal error occurred: Packet content transfer stopped (received 8 bytes)
A fatal error occurred: Packet content transfer stopped (received 8 bytes)
```

I can't seem to find a way to fix this error. The code has been checked and it has no error. I have tried in different computers and it keeps having the same problem. I've even tried different codes but it seems to have the same thing.

Thank you very much!

Have a great day,

Riley Smith

Reply

Sara Santos

April 16, 2023 at 8:26 pm

Hi.
Take a look at this guide: https://randomnerdtutorials.com/esp32-troubleshooting-guide/
Regards,

Reply

Alvaro

July 22, 2021 at 4:10 pm

Hi, nice tips, would you share your code? I think a sophisticated filter may also be good for reliable and stable readings. I also use the JSN SR-04T, but need to place a good filter. One thing important to keep in mind is the US sensor mounting, as there are 2 effects:
1- is the blanking distance (where the sensor is no able to measure, due to pulse&echo measurement required delay) hence you may not be able to obtain very close to sensor readings.
2.- Sensor ringing on tanks when you mount the sensor too tight to tank where the tank may be "resonating" with false readings.

Reply

Ralph McCleery

July 22, 2021 at 11:03 pm

G'day Alvaro,
The current code I'm working with is here https://pastebin.com/4BUHGjyF

NB:
#1 Current device is an ESP-12F on a BoB
#2 To avoid having to change things I have a slide switch for PGM

#3 I use the onboard led to show activity but note its “Active LOW” on the ESP-12F
#4 As stated above I have increased the pulse duration to 20us.

Cheers
Ralph

Reply

Ralph McCleery
July 26, 2021 at 7:40 pm

Apologies, #1 in my original comment is wrong and it wasn’t the code locking up. Further tests have shown it to be the browser itself that has stopped refreshing and just refreshing solves the problem.

Reply

martinius96
July 22, 2021 at 5:30 pm

If somebody want to try real project for monitoring waterwell level using ESP32 with this HC-SR04 module or its waterproof fully compatible JSN-SR04T module with visualisation of datas to web interface, you can do it in my test DEMO project: arduino.clanweb.eu/studna_s_prekladom/?lang=en

There are available source codes for ESP32 for StandBy / StandBy + OTA / Deep Sleep ULP in Arduino Core and also one experimental implementation for ESP-IDF that is using FreeRTOS with tasks.

Reply

Juergen Dietrich
July 22, 2021 at 8:14 pm

In all schematics the sensor is powered by 5V (!) and the trig-pin and the echo-pin are connected directly (!) to GPIO-pins, even though the voltage of the IO-pins of ESP32 (WROOM) must not exceed 3.3V. How is that possible?

Reply

Juergen Dietrich
July 23, 2021 at 9:36 am

supply, but the GPIOs of ESP32 (WROOM) are NOT 5V tolerant, although they are connected directly (!) to the trigger and echo pins of the sensor. I'm a little bit confused: Did these schematics really work?

Reply

Sara Santos

July 23, 2021 at 9:57 am

Hi.
Yes, they work and don't do any harm to the board.
You can try it yourself.
However, you can add a level-shifter, if you prefer.
You can also use 1k0 and 2k2 resistors in series as suggested in one of the comments.
Regards,
Sara

Reply

Daniel Jaramillo

September 10, 2023 at 11:08 pm

Hello Sara and Juergen.
I'm also confused and would like to know why it outputs 3.3V
I've serched the HC-SR04 datasheet and it says indeed the output is 5V, but the oscilloscope reads 3.3V.

Reply

Juergen Dietrich

July 23, 2021 at 7:50 pm

Many thanks Sara
I trusted you and I copied your circuit as shown in your website (even without series resistors or level shifters). And I was very surprised that it actually worked without any damage to the ESP32. Also the measured values seem to be absolutely reliable. Still, I don't understand it, but I'm very grateful because it simplifies my schematic after all.

Reply

Duncan Craig

August 4, 2021 at 6:50 pm

Hello all,
I have been playing with this recently and enjoying it.
I added something useful, which is specifying a timeout in the pulsein command.
That allows one to limit the sensing distance as it then returns a distance of 0 if the echo was not received before the timeout.
The makes the sensor easier to implement when doing object detection and needing to ignore background objects, by effectively specifying the

Kind regards,
South Africa

Reply

Purps

August 4, 2021 at 6:55 pm

Hello, I have been trying for some time to use the RPLIDAR 360 ° laser scanner in such a way that I can capture moving objects (e.g. a bowling ball) in a defined area and evaluate the distance to the scanner. Have you ever dealt with this? I'm grateful for every tip.

Thanks, XXAL

Reply

August 6, 2021 at 11:08 am

Hi.
Unfortunately, we don't have any tutorials on that subject.
Regards,
Sara

Reply

markee suyi
August 26, 2021 at 6:28 am

To prove that I have read this sensor article, leave a footprint!

Reply

Morgan
July 15, 2022 at 5:11 am

the serial monitor, numbers 0.00 on both cm and inch displays. Is there anything I need to change for it the show the actual correct distances?

Reply

Sara Santos

July 16, 2022 at 5:18 pm

Hi.
Double-check the wiring.
Regards,
Sara

Reply

August 17, 2022 at 3:42 pm

Good afternoon: I'm assembling an ESP32 with an HC-SR04 Following the information in your tutorial, I've reached a point where I can't get your code to work. I try to get it to work with Wi-Fi and be able to incorporate it into Home Assistant, or nodered, the ESPHOME dashboard gives an error with // and ;
Can you help me?

Reply

Sara Santos
August 17, 2022 at 11:38 pm

Hi.
You need to be more specific with the error that you're getting.
Otherwise, it's very difficult to understand what might be wrong.
Regards,
Sara

Reply

Enric Peña
August 18, 2022 at 11:29 am

Hello
Well, I have ESPHOME installed on a raspberry pi4 with docker and I

code-Rui-Santos), where ESPHOME shows the syntax errors I don't know how to modify the code so that it works and be able to update the ESP32 with wifi, and see the information in home assistant, or MQTT. How can I send you the files to see the problem?

Reply

Enric Peña

August 18, 2022 at 3:19 pm

Thanks for your information, you can close this post

Reply

Wiesław Sobczak

September 30, 2022 at 5:11 pm

and if I wanted to connect many sensors, for example, as in the car 4 pcs behind and 4 pcs front, each must be connected to a different ESP with its screen? can you connect many sensors to one ESP32? have the screen display the L-S-P (Left-Center-Right) sides and the distance

Reply

Sara Santos

October 1, 2022 at 10:06 am

Yes, you can connect many sensors to the same board.

Reply

Mathéo

October 8, 2022 at 12:00 pm

Sorry but i don’t understand how obtained decimal point with int may you explain me please.

Reply

Mathéo

October 8, 2022 at 12:12 pm

Sorry i see a error in full code you wrote : int distanceCm;
int distanceInch;
But above you have put float.
Have good day and thanks you for your work , i love your work and your example.

Reply

May 10, 2023 at 9:26 pm

Thank you for this amazing tutorial,
I have one question, how can i change the output of distance units on the display (cm in this case) to Percentage % ? as i want measure water level in a tank .
thank you

Reply

Graham
December 10, 2023 at 4:34 pm

THANKS!
My first attempt worked fine as above.

A second and third circuit failed but as soon as I added level shifters all was fine.

So it seems some ESP cards do indeed need level shifters.

Beware!

Reply

Sara Santos
December 14, 2023 at 1:00 pm

Regards,
Sara

Reply

Akshay
January 5, 2024 at 10:00 am

My operating range is around 4-5 meters, SO HC-SR04 Ultrasonic Sensor can do that.

Or if you know any please mentioned

Reply

Prof. K. R. Rao
May 10, 2024 at 2:45 am

Very Nice tutorial
I used HC-SR04 that you shown in the tutorial.
I powered the board (HC-SR04) with 3V3 (Vcc=3.3V) that comes from ESP32 WROOM32 and wired as Echo=GPIO16, Trigger=GPIO17.
It works fine.

Reply

Leave a Comment

Name *

Email *

Website

□ Notify me of follow-up comments by email.

□ Notify me of new posts by email.

Affiliate Disclosure: *Random Nerd Tutorials is a participant in affiliate advertising programs designed to provide a means for us to earn fees by linking to Amazon, eBay, AliExpress, and other sites. We might be compensated for referring traffic and business to these companies.*

SMART HOME with Raspberry Pi, ESP32, and ESP8266 » learn how to build a complete home automation system.

RASPBERRY PI
PICO/PICO W
with MicroPython
Learn Raspberry Pi Pico with MicroPython
The complete getting-started guide to get the most out of the Raspberry Pi Pico/Pico W (RP2040) using MicroPython.
RUI SANTOS & SARA SANTOS
RANDOM NERD TUTORIALS

Learn Raspberry Pi Pico/Pico W with MicroPython » The complete getting started guide to get the most out of the the Raspberry Pi Pico/Pico W (RP2040) microcontroller board using MicroPython programming language.

Learn LVGL
Build GUIs for ESP32 Projects
Compatible with ESP32 Cheap Yellow Display and TFT 2.8 inch ILI9341 Touchscreen Display
LVGL
Learn LVGL: Build GUIs for ESP32 Projects
RUI SANTOS & SARA SANTOS
RANDOM NERD TUTORIALS

Projects » Learn how to build Graphical User Interfaces (GUIs) for ESP32 Projects using LVGL (Light Versatile Graphics Library) with the Arduino IDE.