

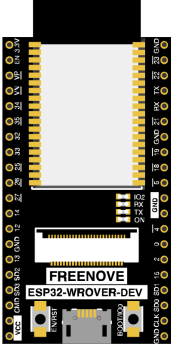
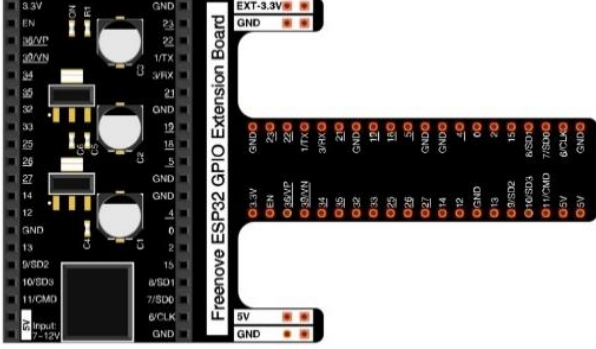
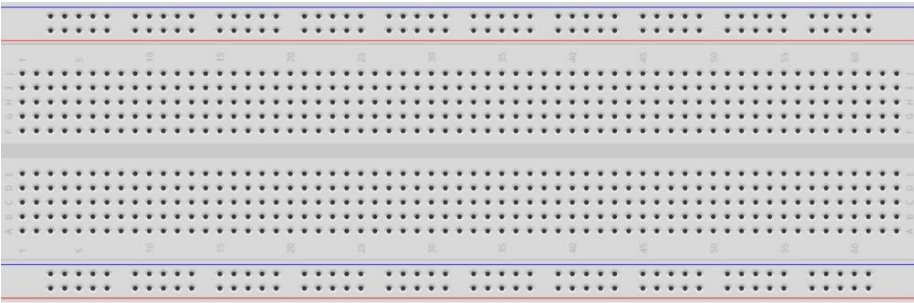


# Chapter 21 Ultrasonic Ranging

In this chapter, we learn a module which use ultrasonic to measure distance, HC SR04.

## Project 21.1 Ultrasonic Ranging

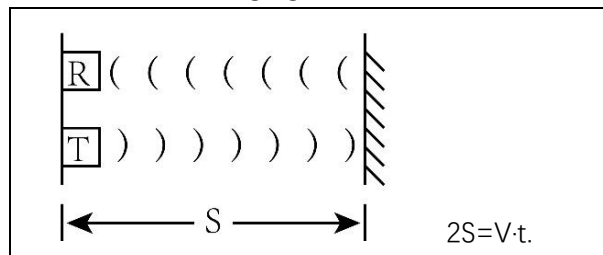
In this project, we use ultrasonic ranging module to measure distance, and print out the data in the terminal.

### Component List

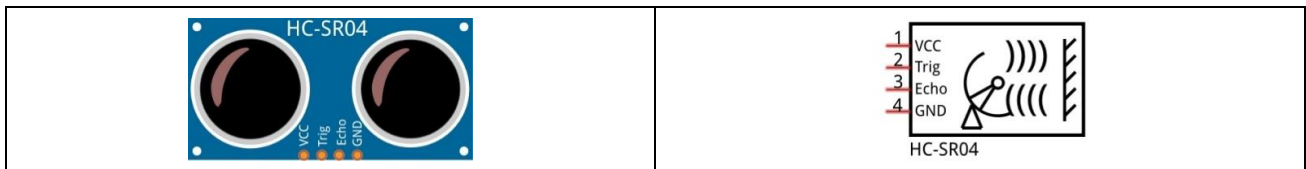
<p>ESP32-WROVER x1</p> 	<p>GPIO Extension Board x1</p> 
<p>Breadboard x1</p> 	
<p>Jumper F/M x4</p> 	<p>HC SR04 x1</p> 

## Component Knowledge

The ultrasonic ranging module uses the principle that ultrasonic waves will be sent back when encounter obstacles. We can measure the distance by counting the time interval between sending and receiving of the ultrasonic waves, and the time difference is the total time of the ultrasonic wave's journey from being transmitted to being received. Because the speed of sound in air is a constant, about  $v=340\text{m/s}$ , we can calculate the distance between the ultrasonic ranging module and the obstacle:  $s=vt/2$ .



The HC-SR04 ultrasonic ranging module integrates both an ultrasonic transmitter and a receiver. The transmitter is used to convert electrical signals (electrical energy) into high frequency (beyond human hearing) sound waves (mechanical energy) and the function of the receiver is opposite of this. The picture and the diagram of the HC SR04 ultrasonic ranging module are shown below:



Pin description:

Pin	Description
VCC	power supply pin
Trig	trigger pin
Echo	Echo pin
GND	GND

### Technical specs:

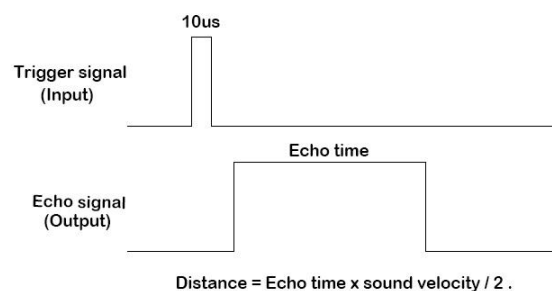
Working voltage: 5V

Working current: 12mA

Minimum measured distance: 2cm

Maximum measured distance: 200cm

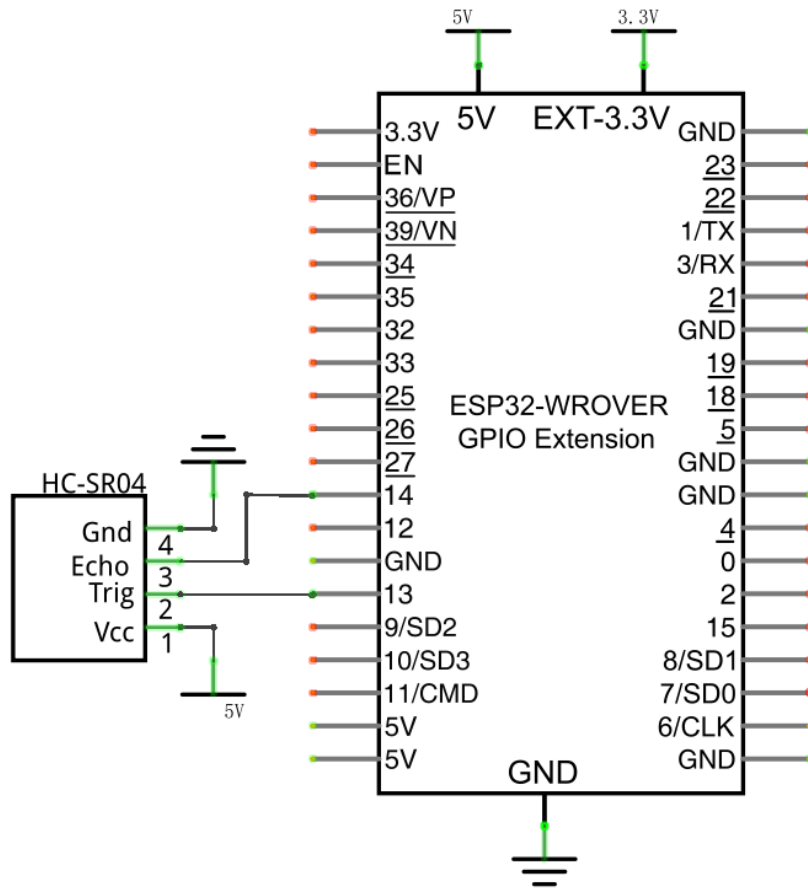
Instructions for use: output a high-level pulse in Trig pin lasting for least 10us, the module begins to transmit ultrasonic waves. At the same time, the Echo pin is pulled up. When the module receives the returned ultrasonic waves from encountering an obstacle, the Echo pin will be pulled down. The duration of high level in the Echo pin is the total time of the ultrasonic wave from transmitting to receiving,  $s=vt/2$ .



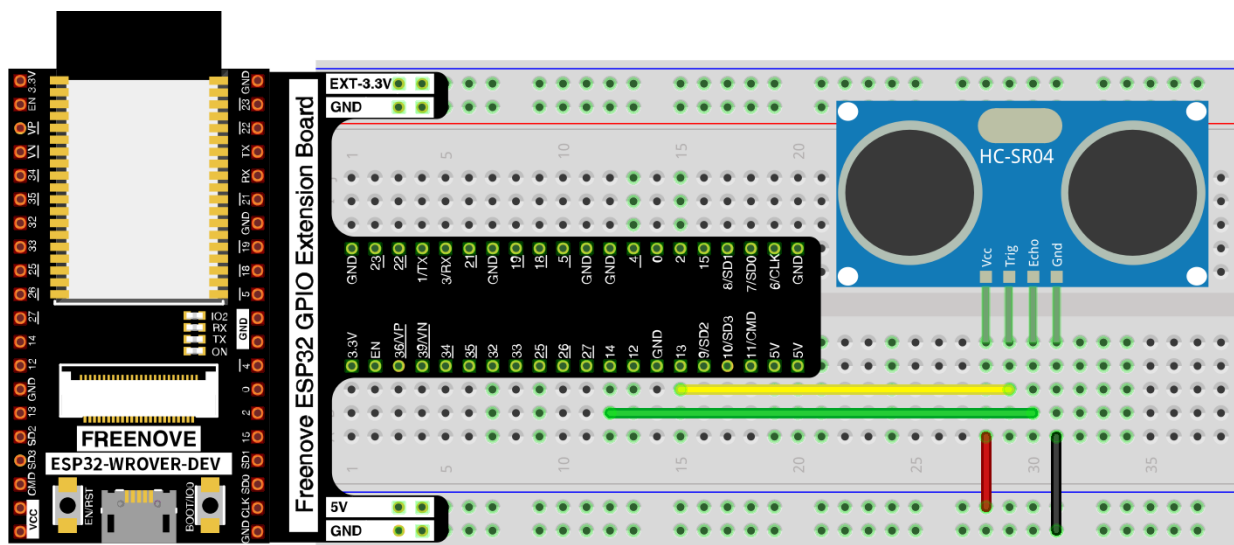
## Circuit

Note that the voltage of ultrasonic module is 5V in the circuit.

Schematic diagram

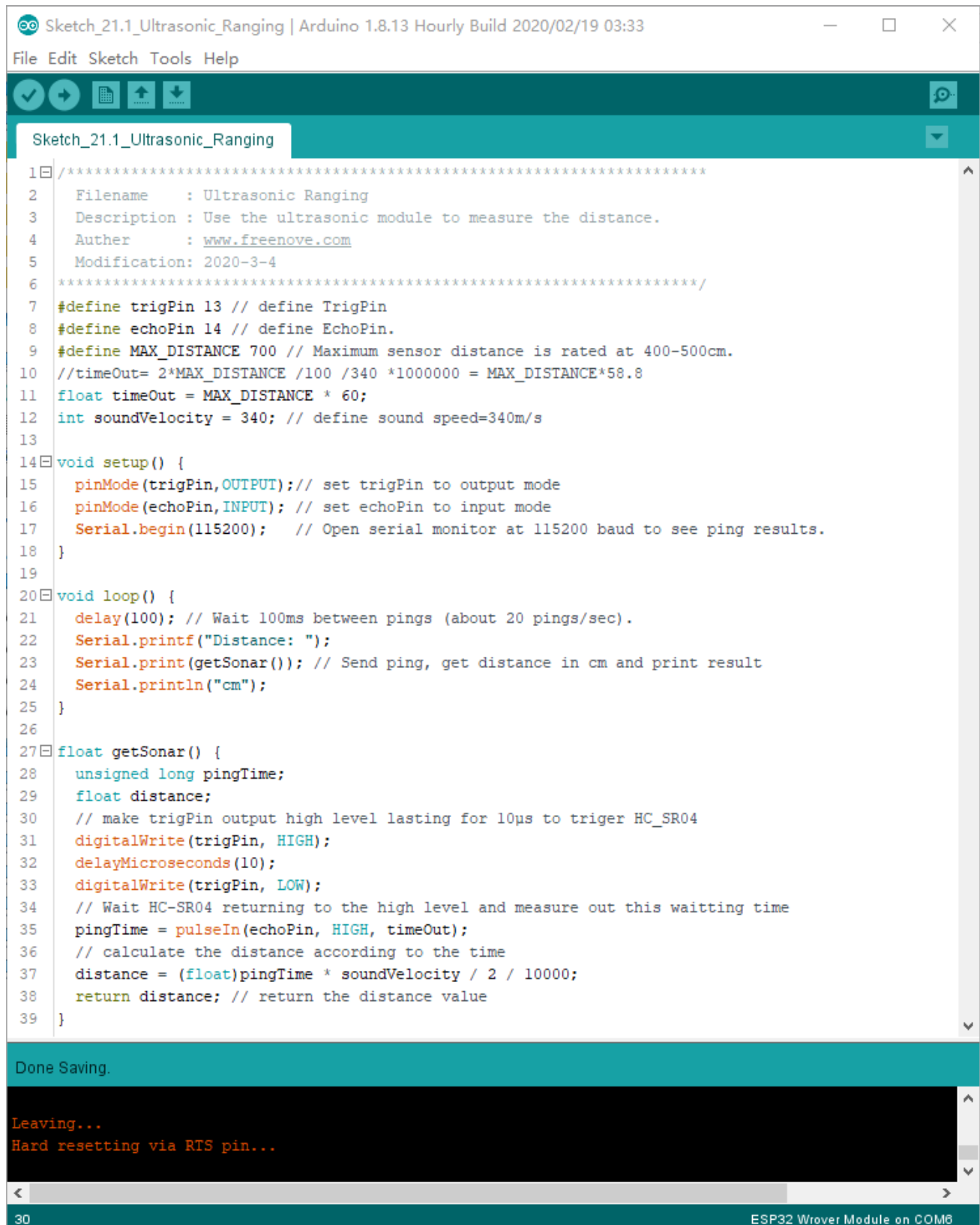


Hardware connection. If you need any support, please feel free to contact us via: [support@freenove.com](mailto:support@freenove.com)



## Sketch

### Sketch\_21.1\_Ultrasonic\_Ranging



```

Sketch_21.1_Ultrasonic_Ranging | Arduino 1.8.13 Hourly Build 2020/02/19 03:33
File Edit Sketch Tools Help

Sketch_21.1_Ultrasonic_Ranging

1 //*****
2   Filename      : Ultrasonic Ranging
3   Description   : Use the ultrasonic module to measure the distance.
4   Author        : www.freenove.com
5   Modification  : 2020-3-4
6   *****/
7   #define trigPin 13 // define TrigPin
8   #define echoPin 14 // define EchoPin.
9   #define MAX_DISTANCE 700 // Maximum sensor distance is rated at 400-500cm.
10  //timeOut= 2*MAX_DISTANCE /100 /340 *1000000 = MAX_DISTANCE*58.8
11  float timeOut = MAX_DISTANCE * 60;
12  int soundVelocity = 340; // define sound speed=340m/s
13
14  void setup() {
15    pinMode(trigPin,OUTPUT); // set trigPin to output mode
16    pinMode(echoPin,INPUT); // set echoPin to input mode
17    Serial.begin(115200); // Open serial monitor at 115200 baud to see ping results.
18  }
19
20  void loop() {
21    delay(100); // Wait 100ms between pings (about 20 pings/sec).
22    Serial.printf("Distance: ");
23    Serial.print(getSonar()); // Send ping, get distance in cm and print result
24    Serial.println("cm");
25  }
26
27  float getSonar() {
28    unsigned long pingTime;
29    float distance;
30    // make trigPin output high level lasting for 10us to trigger HC_SR04
31    digitalWrite(trigPin, HIGH);
32    delayMicroseconds(10);
33    digitalWrite(trigPin, LOW);
34    // Wait HC-SR04 returning to the high level and measure out this waiting time
35    pingTime = pulseIn(echoPin, HIGH, timeOut);
36    // calculate the distance according to the time
37    distance = (float)pingTime * soundVelocity / 2 / 10000;
38    return distance; // return the distance value
39  }

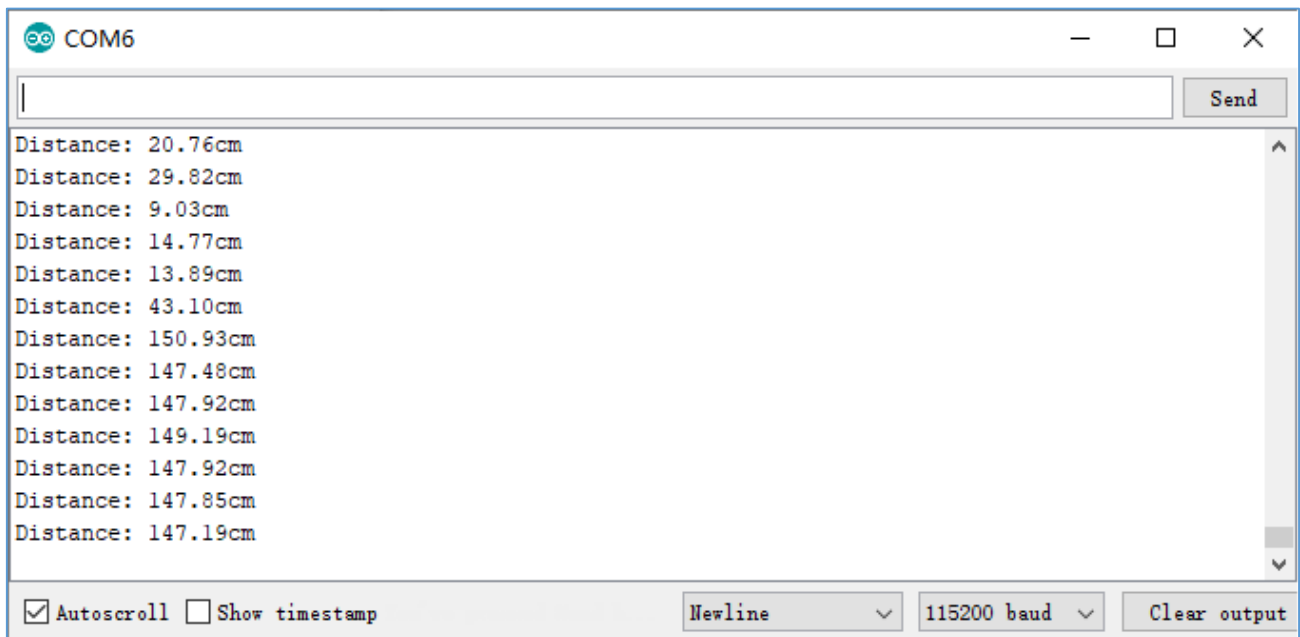
```

Done Saving.

Leaving...  
Hard resetting via RTS pin...

30 ESP32 Wrover Module on COM6

Download the code to ESP32-WROVER, open the serial port monitor, set the baud rate to 115200 and you can use it to measure the distance between the ultrasonic module and the object. As shown in the following figure:



The following is the program code:

```

1  #define trigPin 13 // define trigPin
2  #define echoPin 14 // define echoPin.
3  #define MAX_DISTANCE 700 // Maximum sensor distance is rated at 400-500cm.
4  //timeOut= 2*MAX_DISTANCE /100 /340 *1000000 = MAX_DISTANCE*58.8
5  float timeOut = MAX_DISTANCE * 60;
6  int soundVelocity = 340; // define sound speed=340m/s
7
8  void setup() {
9      pinMode(trigPin, OUTPUT); // set trigPin to output mode
10     pinMode(echoPin, INPUT); // set echoPin to input mode
11     Serial.begin(115200); // Open serial monitor at 115200 baud to see ping results.
12 }
13
14 void loop() {
15     delay(100); // Wait 100ms between pings (about 20 pings/sec).
16     Serial.printf("Distance: ");
17     Serial.print(getSonar()); // Send ping, get distance in cm and print result
18     Serial.println("cm");
19 }
20
21 float getSonar() {
22     unsigned long pingTime;
23     float distance;
24     // make trigPin output high level lasting for 10us to trigger HC_SR04

```

```

25    digitalWrite(trigPin, HIGH);
26    delayMicroseconds(10);
27    digitalWrite(trigPin, LOW);
28    // Wait HC-SR04 returning to the high level and measure out this waiting time
29    pingTime = pulseIn(echoPin, HIGH, timeOut);
30    // calculate the distance according to the time
31    distance = (float)pingTime * soundVelocity / 2 / 10000;
32    return distance; // return the distance value
33 }

```

First, define the pins and the maximum measurement distance.

```

1    #define trigPin 13 // define trigPin
2    #define echoPin 14 // define echoPin.
3    #define MAX_DISTANCE 700 //define the maximum measured distance

```

If the module does not return high level, we cannot wait for this forever, so we need to calculate the time period for the maximum distance, that is, time Out.  $\text{timeOut} = 2 * \text{MAX\_DISTANCE} / 100 / 340 * 1000000$ . The result of the constant part in this formula is approximately 58.8.

```

5    float timeOut = MAX_DISTANCE * 60;

```

Subfunction `getSonar()` function is used to start the ultrasonic module to begin measuring, and return the measured distance in cm units. In this function, first let `trigPin` send 10us high level to start the ultrasonic module. Then use `pulseIn()` to read the ultrasonic module and return the duration time of high level. Finally, the measured distance according to the time is calculated.

```

21    float getSonar() {
22        unsigned long pingTime;
23        float distance;
24        // make trigPin output high level lasting for 10μs to trigger HC_SR04?
25        digitalWrite(trigPin, HIGH);
26        delayMicroseconds(10);
27        digitalWrite(trigPin, LOW);
28        // Wait HC-SR04 returning to the high level and measure out this waiting time
29        pingTime = pulseIn(echoPin, HIGH, timeOut);
30        // calculate the distance according to the time
31        distance = (float)pingTime * soundVelocity / 2 / 10000;
32        return distance; // return the distance value
33    }

```

Lastly, in `loop()` function, get the measurement distance and display it continually.

```

14    void loop() {
15        delay(100); // Wait 100ms between pings (about 20 pings/sec).
16        Serial.printf("Distance: ");
17        Serial.print(getSonar()); // Send ping, get distance in cm and print result
18        Serial.println("cm");
19    }

```

About function `pulseIn()`:

**`int pulseIn(int pin, int level, int timeout);`**

**pin:** the number of the Arduino pin on which you want to read the pulse. Allowed data types: int.

**value:** type of pulse to read: either HIGH or LOW. Allowed data types: int.

**timeout** (optional): the number of microseconds to wait for the pulse to start; default is one second.