

# A01NYUB Waterproof Ultrasonic Sensor Wiki - DFRobot

(<https://www.dfrobot.com/product-1918.html>)

## Introduction

Ultrasonic distance sensor determines the distance to a target by measuring time lapses between the sending and receiving of the ultrasonic pulse. A01NYUB is an waterproof ultrasoinic sensor module with 7.5m effective ranging distance. It is compatible with 3.3V~5V device like Arduino, Raspberry Pi, etc. The average current of A01NYUB is only 15mA so it can be powered by most controllers' IO port.



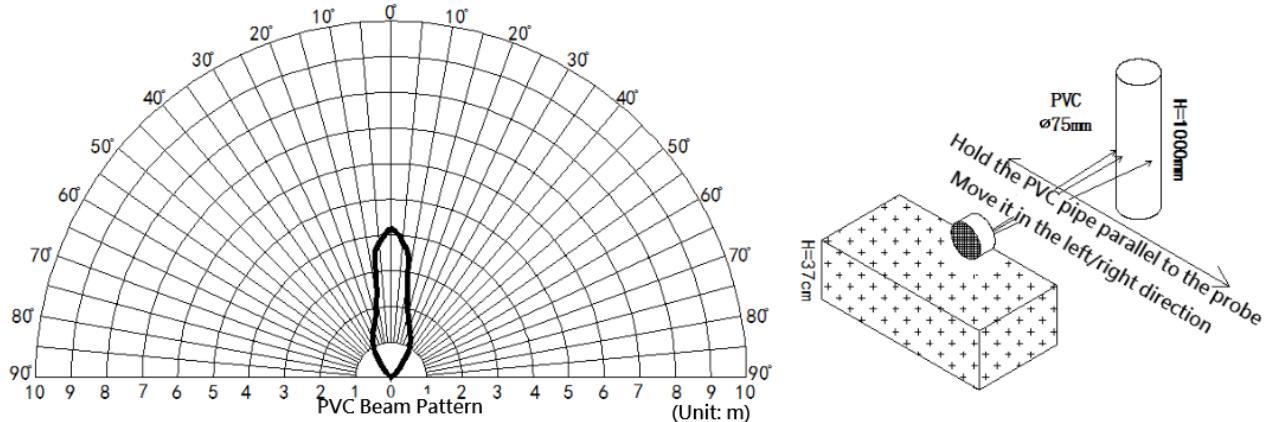
The ultrasonic sensor adopts closed probe of transmitter & receiver, waterproof and dustproof, which could be well suitable for harsh and moist measuring environment. It reserves 2.54-4P interface and adopts UART communication. ME007YS ultrasonic sensor has experienced long-term test and constant optimization so it can offer a pretty fast response time, high stability and sensitivity, and low power consumption.

Use the sensor with Arduino controller to build up your projects, such as backing car annunciator, obstacle avoidance robot, object approaching detection etc.

## Specification

- Operating Voltage: 3.3V-5v
- Average Current: ≤15mA
- Blind Zone Distance: 0-28cm
- Ranging Distance for Flat Object: 28-750cm
- distance resolution: 1mm
- Output: UART

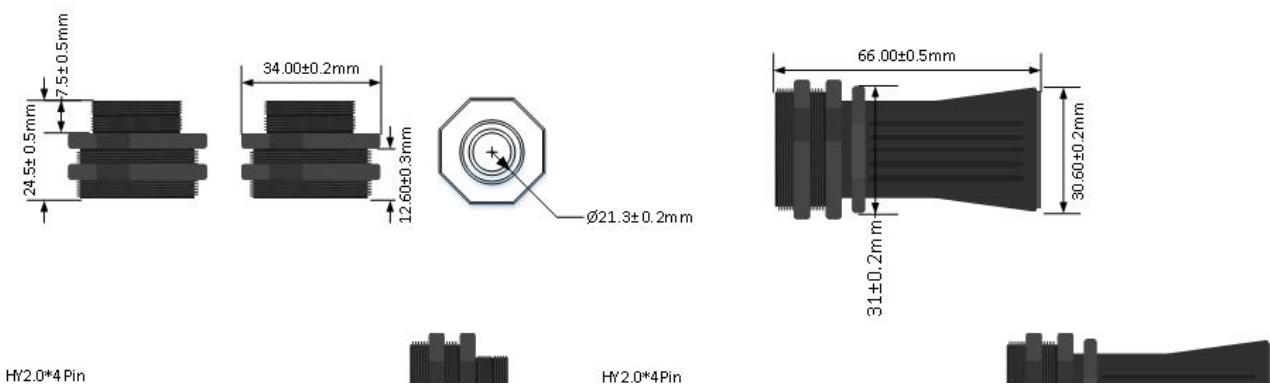
- Response Time: 100ms
- Probe Center Frequency: 40K±1.0K
- Operating Temperature: -15~60°C
- Storage Temperature: -25~80°C
- Protection Rate: IP67

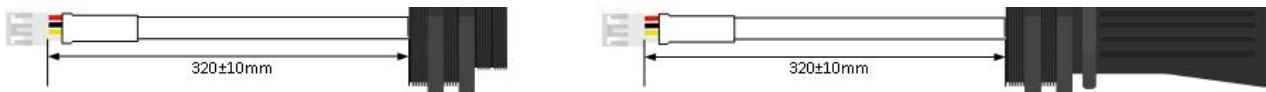


## Features

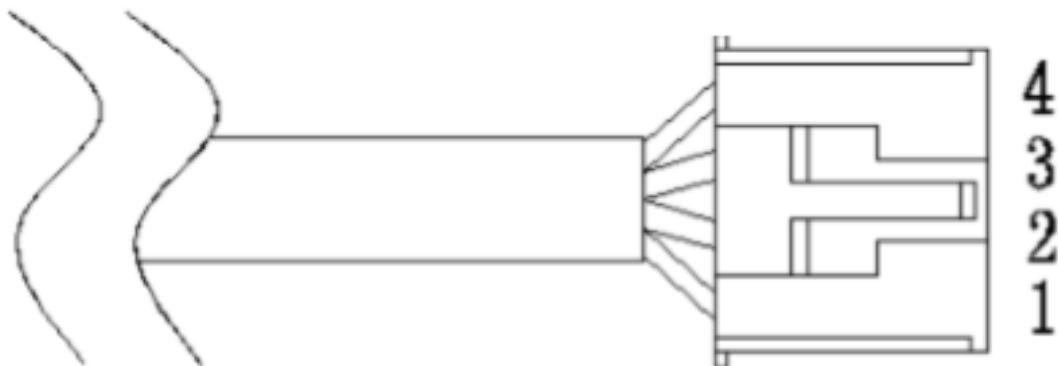
- High Protection Rate
- Strong Resistance
- Stable Output
- Low Power
- Fast Response
- High Antistatic Performance
- Wide Operating Temperature
- High Accuracy
- Small in Size

## Installation Dimension





## Pinout



Label	Name	Description
1	VCC	Power Input
2	GND	Ground
3	RX	Processed Value/Real-time Value Output Selection
4	TX	UART Output

## UART Output

### Output Communication

When "RX" floats or input High level, the module outputs processed value, the data is more steady, response time: 150-300ms; when input Low level, the module outputs real-time value, response time: 150ms.

UART	Data bit	Stop bit	Parity	Band rate
TTL level	8	1	none	9600bps

### UART Output Form

Frame Data	Description	Byte
Header	0xFF	1 byte

DATA_H	Distance Data High 8-bits	1 byte
DATA_L	Distance Data Low 8-bits	1 byte
SUM	Checksum	1 byte

## UART Output

Header	DATA_H	DATA_L	SUM
0xFF	0x07	0xA1	0xA7

Note: checksum only reserves the low 8-bits of the accumulated value.

```

SUM=(Header+Data_H+Data_L)&0x00FF
=(0XFF + 0X07 + 0XA1)&0x00FF
=0XA7;

```

Distance= Data\_H\*256+ Data\_L=0X07A1;

Equal to 1953 when converted into decimal;

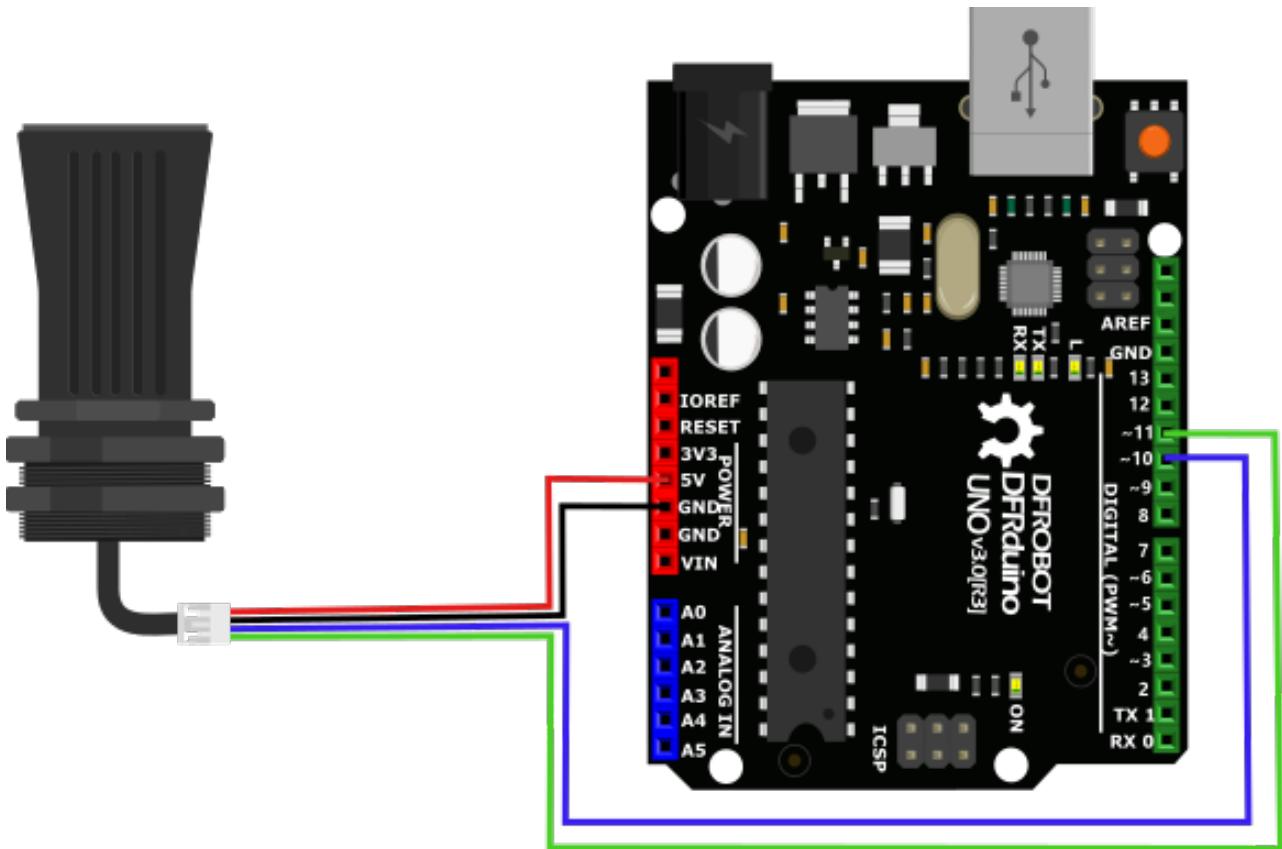
Represent the current measured distance is 1953mm.

## Arduino Platform

### Preparation

- Arduino UNO
- UNO IO Sensor Expansion Board
- A01NYUB Ultrasonic Sensor
- 4P Connector

### Connection



**Sample Code**

```
/*
 @File  : DFRobot_Distance_A01.ino
 @Brief : This example use A01NYUB ultrasonic sensor to measure distance
           With initialization completed, We can get distance value
 @Copyright [DFRobot](https://www.dfrobot.com),2016
           GUN Lesser General Public License
 @version V1.0
 @data  2019-8-28
 */

#include <SoftwareSerial.h>

SoftwareSerial mySerial(11, 10); // RX, TX
unsigned char data[4] = {};
float distance;

void setup()
{
    Serial.begin(57600);
    mySerial.begin(9600);
}

void loop()
{
    do {
        for (int i = 0; i < 4; i++)
        {
            data[i] = mySerial.read();
        }
    } while (mySerial.read() == 0xff);

    mySerial.flush();

    if (data[0] == 0xff)
    {
        int sum;
        sum = (data[0] + data[1] + data[2]) & 0x00FF;
        if (sum == data[3])
        {
            distance = (data[1] << 8) + data[2];
            if (distance > 280)
            {
                Serial.print("distance=");
                Serial.print(distance / 10);
                Serial.println("cm");
            } else

```

```

    {
        Serial.println("Below the lower limit");
    }
} else Serial.println("ERROR");
}

delay(150);
}

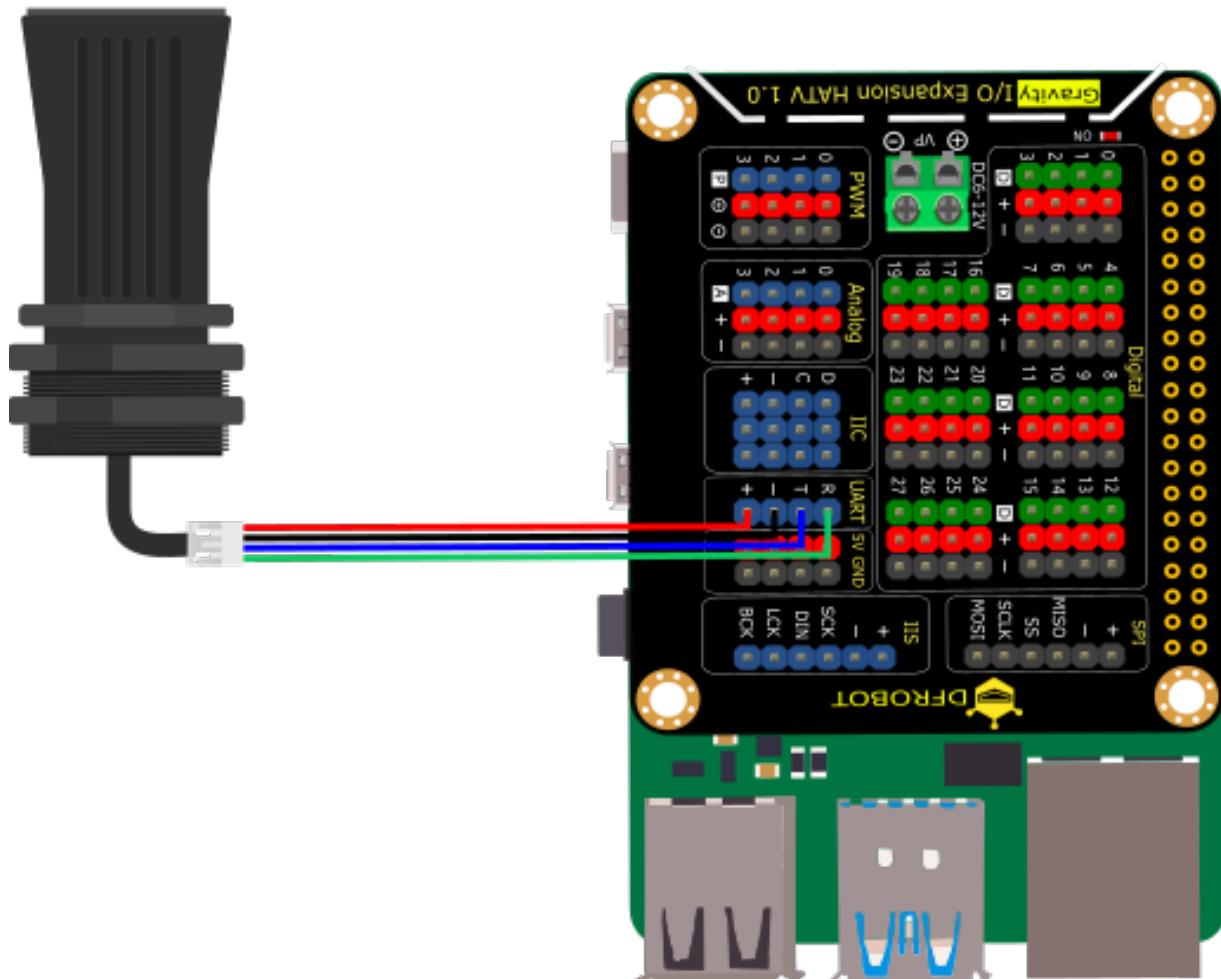
```

## Raspberry Pi Platform

### Preparation

- Raspberry Pi 4B
- Raspberry Pi IO Expansion Board
- A01NYUB Ultrasonic Sensor
- 4P Connector

### Raspberry Pi Connection



### Sample Code

Download the Ultrasonic Sensor Library ([https://github.com/DFRobot/DFRobot\\_RaspberryPi\\_A02YYUW](https://github.com/DFRobot/DFRobot_RaspberryPi_A02YYUW))

```
# -*- coding:utf-8 -*-

'''!
@file DFRobot_RaspberryPi_A02YYUW.py
@brief Ranging distance sensor.
@copyright Copyright (c) 2010 DFRobot Co.Ltd (http://www.dfrobot.com)
@license The MIT License (MIT)
@author Arya(xue.peng@dfrobot.com)
@version V1.0
@date 2021-08-30
@url https://github.com/DFRobot/DFRobot\_RaspberryPi\_A02YYUW
'''

import serial

import time

class DFRobot_A02_Distance:

    ## Board status
    STA_OK = 0x00
    STA_ERR_CHECKSUM = 0x01
    STA_ERR_SERIAL = 0x02
    STA_ERR_CHECK_OUT_LIMIT = 0x03
    STA_ERR_CHECK_LOW_LIMIT = 0x04
    STA_ERR_DATA = 0x05

    ## last operate status, users can use this variable to determine the result
    last_operate_status = STA_OK

    ## variable
    distance = 0

    ## Maximum range
    distance_max = 4500
    distance_min = 0
    range_max = 4500

    def __init__(self):
        '''
        @brief Sensor initialization.
        '''

        self._ser = serial.Serial("/dev/ttyAMA0", 9600)
        if self._ser.isOpen() != True:
            self.last_operate_status = self.STA_ERR_SERIAL
```

```
def set_dis_range(self, min, max):
    self.distance_max = max
    self.distance_min = min

def getDistance(self):
    """
        @brief      Get measured distance
        @return     measured distance
    """
    self._measure()
    return self.distance

def _check_sum(self, l):
    return (l[0] + l[1] + l[2])&0x00ff

def _measure(self):
    data = [0]*4
    i = 0
    timenow = time.time()

    while (self._ser.inWaiting() < 4):
        time.sleep(0.01)
        if ((time.time() - timenow) > 1):
            break

    rlt = self._ser.read(self._ser.inWaiting())
    #print(rlt)

    index = len(rlt)
    if(len(rlt) >= 4):
        index = len(rlt) - 4
        while True:
            try:
                data[0] = ord(rlt[index])
            except:
                data[0] = rlt[index]
            if(data[0] == 0xFF):
                break
            elif (index > 0):
                index = index - 1
            else:
                break
        #print(data)
        if (data[0] == 0xFF):
            try:
                data[1] = ord(rlt[index + 1])
                data[2] = ord(rlt[index + 2])
                data[3] = ord(rlt[index + 3])
            except:
                data[1] = rlt[index + 1]
                data[2] = rlt[index + 2]
                data[3] = rlt[index + 3]
```

```
i = 4
#print(data)
if i == 4:
    sum = self._check_sum(data)
    if sum != data[3]:
        self.last_operate_status = self.STA_ERR_CHECKSUM
    else:
        self.distance = data[1]*256 + data[2]
        self.last_operate_status = self.STA_OK
    if self.distance > self.distance_max:
        self.last_operate_status = self.STA_ERR_CHECK_OUT_LIMIT
        self.distance = self.distance_max
    elif self.distance < self.distance_min:
        self.last_operate_status = self.STA_ERR_CHECK_LOW_LIMIT
        self.distance = self.distance_min
    else:
        self.last_operate_status = self.STA_ERR_DATA
return self.distance
```

## FAQ

For any questions, advice or cool ideas to share, please visit the **DFRobot Forum** (<https://www.dfrobot.com/forum>)

## More Documents

- STP Model (<https://dfimg.dfrobot.com/62b2fb5caa613609f271523c/wiki/622e26575da60e887776c26a41a8a1fb.rar>)

**Back to Top** 