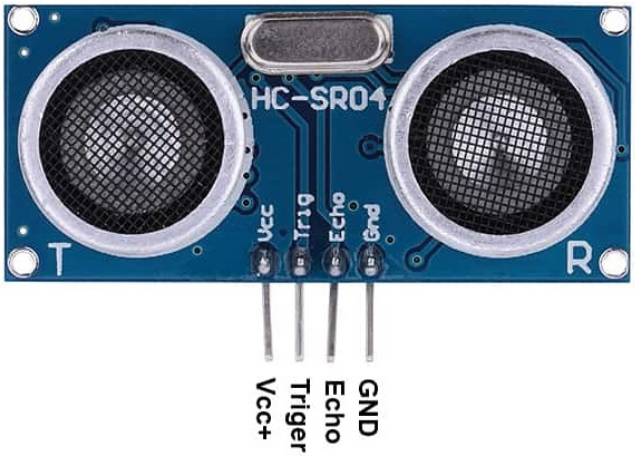
HC-SR04

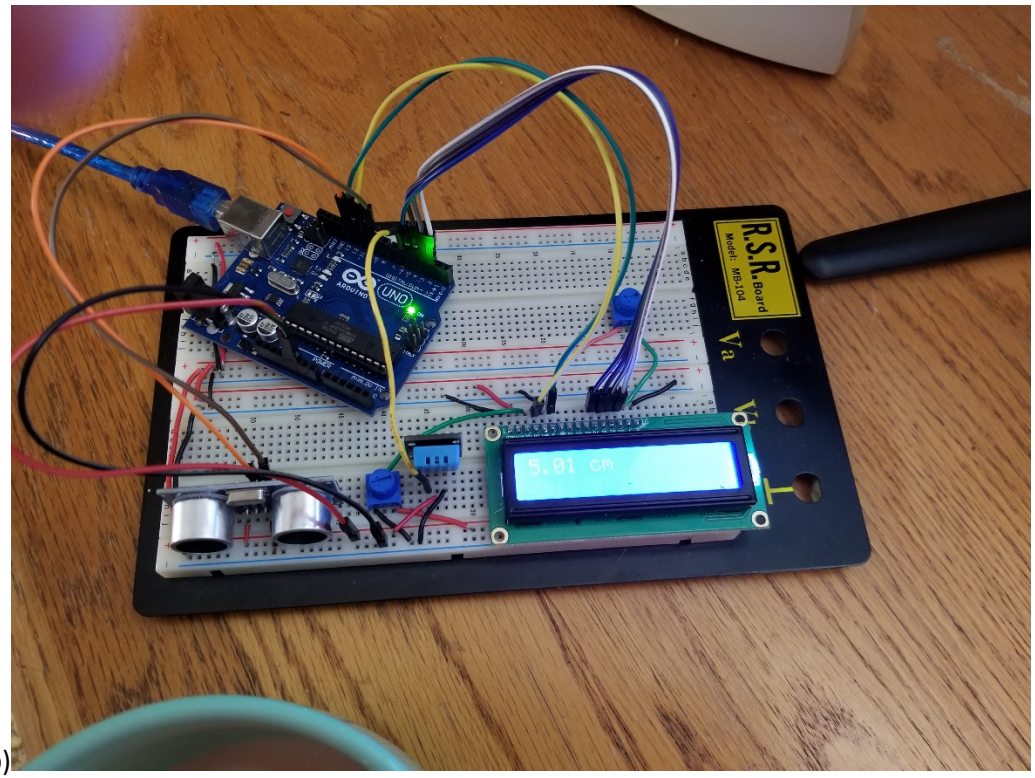
Ultrasonic Distance Sensor

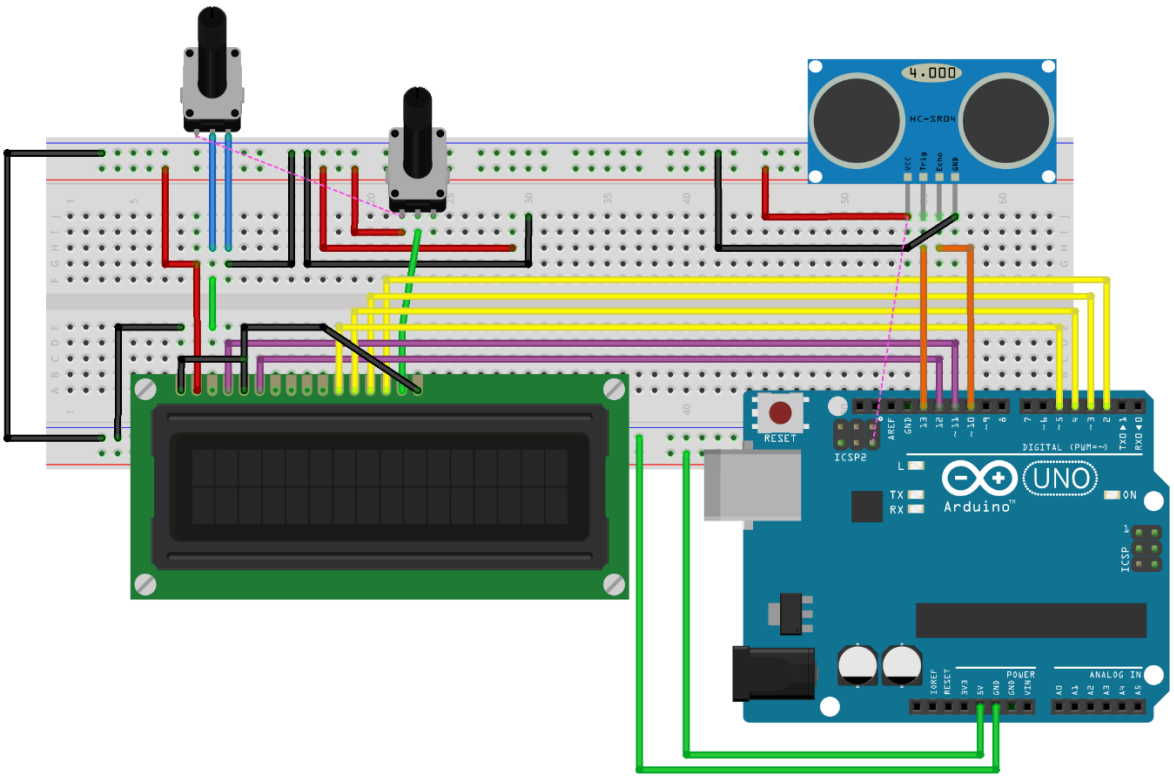
On an Arduino



(a)

(b)





(c)

(d)

COM4

Send

```
Sound: 344.51 m/s, Humid: 26.00 %, Temp: 21.10 C, Distance: 5.01 cm
Sound: 344.57 m/s, Humid: 26.00 %, Temp: 21.20 C, Distance: 5.01 cm
Sound: 344.57 m/s, Humid: 26.00 %, Temp: 21.20 C, Distance: 5.08 cm
Sound: 344.51 m/s, Humid: 26.00 %, Temp: 21.10 C, Distance: 5.01 cm
Sound: 344.57 m/s, Humid: 26.00 %, Temp: 21.20 C, Distance: 5.01 cm
Sound: 344.57 m/s, Humid: 26.00 %, Temp: 21.20 C, Distance: 5.01 cm
Sound: 344.57 m/s, Humid: 26.00 %, Temp: 21.20 C, Distance: 5.01 cm
Sound: 344.57 m/s, Humid: 26.00 %, Temp: 21.20 C, Distance: 5.01 cm
Sound: 344.57 m/s, Humid: 26.00 %, Temp: 21.20 C, Distance: 5.08 cm
Sound: 344.57 m/s, Humid: 26.00 %, Temp: 21.20 C, Distance: 5.01 cm
Sound: 344.57 m/s, Humid: 26.00 %, Temp: 21.20 C, Distance: 5.08 cm
Sound: 344.57 m/s, Humid: 26.00 %, Temp: 21.20 C, Distance: 5.08 cm
Sound: 344.57 m/s, Humid: 26.00 %, Temp: 21.20 C, Distance: 5.01 cm
Sound: 344.57 m/s, Humid: 26.00 %, Temp: 21.20 C, Distance: 5.01 cm
Sound: 344.57 m/s, Humid: 26.00 %, Temp: 21.20 C, Distance: 5.08 cm
Sound: 344.57 m/s, Humid: 26.00 %, Temp: 21.20 C, Distance: 5.08 cm
```

Autoscroll Show timestamp Newline 9600 baud Clear output

Fig 1: (a) HC-SR04 Sensor, (b) Circuit used to test the ultrasonic sensor, (c) Fritzing Circuit Diagram, (d) Serial monitor display of the acquired data

1. Introduction

The HC-SR04 is the most popular, economical, easy to use ultrasonic distance sensor available to the general public and hobbyists. its range extends from 2 cm to 400 cm. The most commonly usage of this sensor is in obstacle-avoidance projects.

HC-SR04 Specifications

Operating voltage	5 V
Operating current	15 mA
Operating Frequency	40 kHz
Measuring range	2 – 400 cm
Resolution (accuracy)	3 mm
Measuring angle	15 degrees
Trigger input signal	10 µs high pulse
Dimension	45 x 20 x 15mm

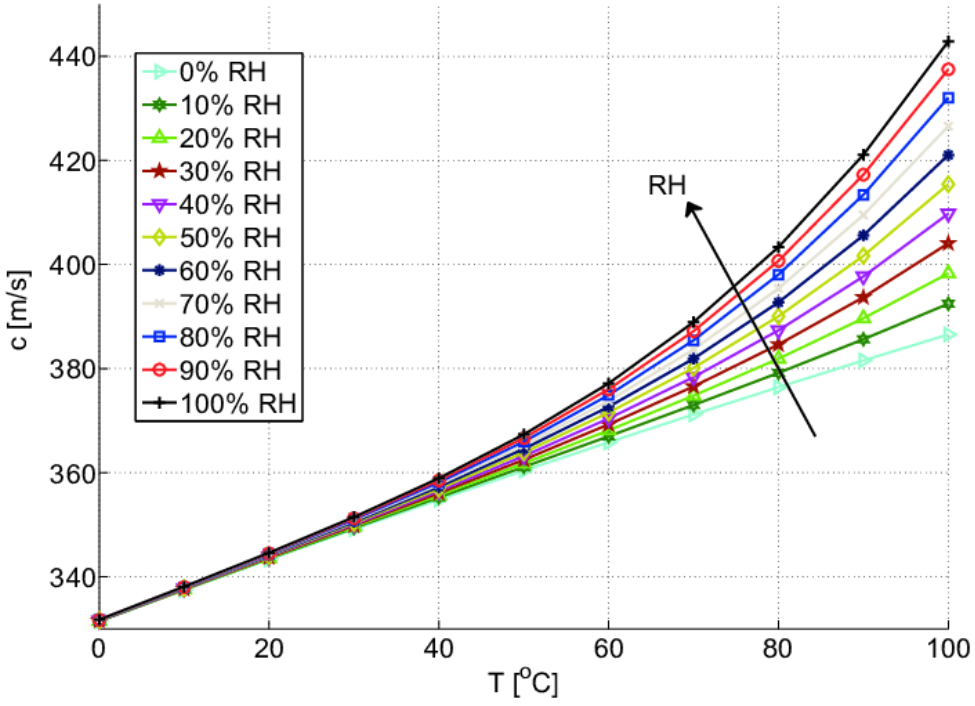


Fig 2: Speed of sound vs. temperature and relative humidity according to [2], p = 101.3 kP a, 314 ppm CO 2

High Accuracy Acoustic Relative Humidity Measurement in Duct Flow with Air - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Speed-of-sound-vs-temperature-and-relative-humidity-according-to-2-p-1013-kP-a_fig9_51873087 [accessed 10 Jan, 2021]

We can see that the humidity has very little to no effect on the speed of sound as long as we operate in the range 0-40˚C. beyond 60˚C, we may not want to ignore the variation in the speed of sound. However, for any level of humidity, there is a corrective term to be included in the relationship between the speed of sound and the temperature, where at 0˚C, the speed of sound is 331.3 m/s.

Hence, for more accurate measurements of the speed of sound, one may want to use the following expression

V = 331.3 + (0.606 × T)

V = Speed of sound (m/s)
T = Air Temperature (°C)

Difficulties

- Targets cannot be more than 4m away from the sensor
- If the sensor is not facing the target, there could be the possibility that there will be no reflected wave reaching the sensor, and you may even get a measured reflection off the floor
- There will also be no measurable reflection if the target is too small
- If the target is soft, or has irregular contours, the measurements will be inaccurate as the target may even absorb some of the energy in the sound waves

Description of the process and evaluations of desired parameters

The start of the process requires the triggering of a 10-µs duration TTL rectangular pulse. When that is done, the sensor will transmit a 40-kHz burst of pulses (sound waves, which at this frequency is beyond the hearing range of a human being).

When the reflected signal is received, the Echo pin produces a pulse whose duration is proportional to the time between the time the burst was generated, and the time the reflected wave was received.

Code

```
/*

  HC-SR04 with Temp and Humidity Demonstration

  Demonstrates range measurements through the HC-SR04 Ultrasonic Range Finder
```

```
  enhanced by the availability of the temperature and humidity through DHT11
  Displays results on Serial Monitor and LCD display
 */
 // include libraries for the DHT11, HC-SR04, and the 1602A QAPASS display
#include "DHT.h"  // helps interface with the DHT11 (valid also for DHT22)
#include "NewPing.h"  // provides useful function to interact with the HC-SR04
#include "LiquidCrystal.h"  // valid for most 16x2 LCD display

// Define Constants

#define DHTpin 6       // DHT-11 Signal pin connected to digital pin 6
#define DHTtype DHT11   // DHT Type is DHT11
#define Trigger_pin  13  // trigger pin of HC-SR04 connected to digital pin 13
#define Echo_pin     10   // Echo pin of HC-SR04 connected to digital pin 10
#define Max_distance 400
 // There was a problem when the echo pin was connected to digital pin 13
// (possibly some interference with another Arduino internal component connected to it)

// Initialize the HC-SR04 Ultrasonic Distance Sensor
NewPing sonar(Trigger_pin, Echo_pin, Max_distance);

// Define Variables
```

```
const int rs = 11, en = 12, d4 = 5, d5 = 4, d6 = 3, d7 = 2;

// Initialize the 1602A QAPASS LCD display
LiquidCrystal ECE114(rs, en, d4, d5, d6, d7);

// Initialize DHT sensor
DHT dht(DHTpin, DHTtype);

float hum;    // Stores humidity value in percent
float temp;   // Stores temperature value in ˚Celcius
float duration; // Stores HC-SR04 pulse duration value
float distance; // Stores calculated distance in cm
float soundsp;  // Stores calculated speed of sound in m/s
float soundcm;  // Stores calculated speed of sound in cm/us
int iterations = 5;  // in this case, we chose 5 runs of measurements to be averaged

void setup() {
  Serial.begin (9600);  // initialize the serial monitor
  dht.begin();  // get the DHT11 to be ready for measurements

  ECE114.begin(16, 2);
}
```

```
void loop()
{
  ECE114.clear();  //clear LCD display
  ECE114.setCursor(0, 0); // move cursor to first row, first character location
  delay(1000);  // Delay by 1 second so DHT11 sensor can stabilize

  hum = dht.readHumidity();  // Get Humidity value
  temp= dht.readTemperature();  // Get Temperature value

   // Calculate the Speed of Sound in m/s
   // the speed of sound 343 m/s is valid for a temperature of 19˚C
   // soundsp = 331.4 + (0.606 * temp) + (0.0124 * hum);
   soundsp = 331.4 + (0.606 * temp);
   // If you need to involve humidity in the calculation of the speed of sound
   // replace the soundsp expression with the previous line
    // Convert to cm/us, cm = centimeter = 1m/100, us = microsecond = 1e-6s
    soundcm = soundsp*100 / 1000000; // speed of sound in cm/us

    duration = sonar.ping_median(iterations);
    // Calculate the distance
    distance = (duration / 2) * soundcm;

    // Send results to Serial Monitor
```

```
Serial.print("Sound: ");
Serial.print(soundsp);
Serial.print(" m/s, ");
Serial.print("Humid: ");
Serial.print(hum);
Serial.print(" %, Temp: ");
Serial.print(temp);
Serial.print(" C, ");
Serial.print("Distance: ");

if (distance >= 400 || distance <= 2) {
Serial.print("Out of range");
}
else {
Serial.print(distance);
Serial.print(" cm");
delay(500);
}
ECE114.print(distance);
ECE114.print(" cm");
 ECE114.setCursor(0, 1);
ECE114.print("Temp: ");
```

```
   ECE114.print(temp);

   ECE114.print(" " "\xDF" "C"); // \xDF degree symbol LCD

   delay(2000);  Serial.println(" ");

}
```

As was mentioned in the code, the expression giving the speed of sound for any temperature and humidity level is:

Speed of Sound = 331.4 + (0.606 * temp) + (0.0124 * hum)

However, from the previous graph given in Fig 2, as long as the temperature is less than 40˚C, the humidity variations can be ignored. Hence we used

Speed of Sound = 331.4 + (0.606 * temp)

The speed of sound given as 343 m/s is valid only at the temperature of 19˚C.

Since the waves travel from the sensor to the target and back to the sensor, the duration of the generated pulse corresponds to twice the distance from the sensor to the target, hence the need for a division by 2.

Distance (cm) = Speed of sound (cm/µs) × Time (µs) / 2

The NewPing library allows us to set a max distance to read, as in this case where the sensor has been specified to work properly when the distance does not exceed 400 cm. In addition, it has a built-in median filter that yields the average of a datum, in this case the duration, over many iterations. This library could have been used to evaluate the distance directly through

`distance = sonar.ping_cm()` // measurement of distance in centimeters, or

`distance = sonar.ping_in()` // measurement of distance in inches

Unfortunately, in both cases, the result will be an integer, which could be useful in some cases but not others.

In the absence of the NewPing library, and using only the Arduino library, see how more complex the coding would be:

```
// Define Trig and Echo pin:

#define trigPin 13
```

```
#define echoPin 10
// Define variables:
long duration;
int distance;

void setup() {
// Define inputs and outputs:
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
//Begin Serial communication at a baudrate of 9600:
Serial.begin(9600);
}

void loop() {

// Create the 10-us rectangular pulse to trigger the HC-SR04
// to make sure we start with a low-to-high transition, we force the pulse to be low for 2 us
digitalWrite(trigPin, LOW); // Clear the trigPin by setting it LOW:
delayMicroseconds(2);
// Trigger the sensor by setting the trigPin high for 10 microseconds:
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);

// Read the echoPin, pulseIn() returns the duration (length of the pulse) in microseconds:
duration = pulseIn(echoPin, HIGH);
// Calculate the distance, assuming the speed of sound valid for 19˚C
distance = duration * 0.034 / 2;
// Print the distance on the Serial Monitor
```

```
Serial.print("Distance = ");

Serial.print(distance);

Serial.println(" cm");

delay(1000);

}
```

Measurements can be taken faster than the 1 second that has been chosen (delay(1000)) if needed. It is obvious that the limitation will be related to the generation of the triggering pulse, the burst of 40kHz pulses, the travel time, and the time to process the data. All instructions are from the Arduino function set only. This is the main reason we appreciate the usage of the libraries. Not only will the code be simpler, and shorter, we also have access to data that the Arduino instruction set may not make available readily. Note that since the distance has been defined as an integer, the results will be given as integers in cm in this case.

NewPing Library

Assuming

```
NewPing sonar(12, 11, 200);
```

This initializes NewPing to use pin 12 for trigger output, pin 11 for echo input, with a maximum ping distance of 200cm. max_cm_distance is optional (default = 500cm). If connecting using a single pin, specify the same pin for both trigger_pin and echo_pin as the same pin is doing both functions.

- **sonar.ping([max_cm_distance])** - Send a ping and get the echo time (in microseconds) as a result. [max_cm_distance] allows you to optionally set a new max distance.
- **sonar.ping_in([max_cm_distance])** - Send a ping and get the distance in whole inches. [max_cm_distance] allows you to optionally set a new max distance.
- **sonar.ping_cm([max_cm_distance])** - Send a ping and get the distance in whole centimeters. [max_cm_distance] allows you to optionally set a new max distance.
- **sonar.ping_median(iterations [, max_cm_distance])** - Do multiple pings (default=5), discard out of range pings and return median in microseconds. [max_cm_distance] allows you to optionally set a new max distance.
- **sonar.convert_in(echoTime)** - Convert echoTime from microseconds to inches.
- **sonar.convert_cm(echoTime)** - Convert echoTime from microseconds to centimeters.
- **sonar.ping_timer(function [, max_cm_distance])** - Send a ping and call function to test if ping is complete. [max_cm_distance] allows you to optionally set a new max distance.
- **sonar.check_timer()** - Check if ping has returned within the set distance limit.
- **NewPing::timer_us(frequency, function)** - Call function every frequency microseconds.
- **NewPing::timer_ms(frequency, function)** - Call function every frequency milliseconds.
- **NewPing::timer_stop()** - Stop the timer.

```
// ---------------------------------------------------------------------------
-
// Example NewPing library sketch that pings 3 sensors 20 times a second.
// ---------------------------------------------------------------------------
-

#include <NewPing.h>

#define SONAR_NUM 3      // Number of sensors.
#define MAX_DISTANCE 200 // Maximum distance (in cm) to ping.

NewPing sonar[SONAR_NUM] = {   // Sensor object array.
  NewPing(4, 5, MAX_DISTANCE), // Each sensor's trigger pin, echo pin, and
max distance to ping.
  NewPing(6, 7, MAX_DISTANCE),
  NewPing(8, 9, MAX_DISTANCE)
};

void setup() {
```

```
  Serial.begin(115200); // Open serial monitor at 115200 baud to see ping
results.
}

void loop() {
  for (uint8_t i = 0; i < SONAR_NUM; i++) { // Loop through each sensor and
display results.
    delay(50); // Wait 50ms between pings (about 20 pings/sec). 29ms should
be the shortest delay between pings.
    Serial.print(i);
    Serial.print("=");
    Serial.print(sonar[i].ping_cm());
    Serial.print("cm ");
  }
  Serial.println();
}
```