PAPERS | APRIL 01 2022

An Arduino Investigation of the Temperature Dependence of the Speed of Sound in Air

Calin Galeriu

Phys. Teach. 60, 284–288 (2022)
https://doi.org/10.1119/10.0009993

An Arduino Investigation of the Temperature Dependence of the Speed of Sound in Air

Calin Galeriu, Mark Twain International School, Bucharest, Romania

The determination of the speed of sound in air is a classical experiment, usually performed with a resonance tube apparatus. The measured value can be checked against Eq. (1), which describes the temperature dependence of the speed of sound in dry air. A modern implementation of this speed of sound investigation[1,2] uses an Arduino Uno microcontroller board, an HC-SR04 ultrasonic distance sensor, and a DS18B20 temperature sensor. The distance sensor's transmitter produces a burst of eight ultrasonic rectangular pulses that travel through the air, reflect on an object placed in front at a distance d, and then return to the sensor's receiver after an echo time t. This Arduino investigation, unfortunately, is harder to perform than one might expect after a first reading of Ref. 1 or 2. In this article we discuss some sources of experimental errors that can complicate this laboratory activity, and we describe some important steps that must be included in the data collection and analysis procedure, in order to obtain successful results every time.

The theoretical speed of sound in dry air is given by the formula[3]

$$v = 20.05\sqrt{T_K}\ \text{m/s} = 20.05\sqrt{273.15 + T_C}\ \text{m/s}, \qquad (1)$$

where T_K is the temperature in kelvins, and T_C is the temperature in degrees Celsius. A first order approximation of Eq. (1) gives[3]

$$v \approx (\,331.4 + 0.6\ T_C\,)\ \text{m/s}. \qquad (2)$$

The experimental speed of sound is determined as

$$v = \frac{2d}{t}, \qquad (3)$$

where d is the distance traveled by the ultrasonic wave from the transmitter to the wall, equal to the distance traveled from the wall back to the receiver. In a typical experiment the temperature might change from about 10 °C to about 20 °C, and the speed of sound changes from about 337 m/s to about 343 m/s. The relative variation of the speed is less than 2%, and therefore the distance d and the echo time t have to be measured with great accuracy.

The experimental errors related to the HC-SR04 sensor

What can we say about the error in the measurement of the distance d? First of all, the HC-SR04 datasheet[4] is not very clear about how the distance d should be measured. The HC-SR04 sensor rises about 12 mm above the printed circuit board, as seen in Fig. 1, with a transmitter and a receiver of cylindrical shape. In the absence of a clear mark on the sensor, one should start by finding the zero position, like in a calibration procedure, by placing the sensor in front of a wall and matching the reading from the sensor to the reading on a ruler. This zero position is located somewhere in the middle, where the physical transmitter and receiver are located inside the cylindrical structures. It turns out that we don't have to actually trace this line (the zero mark) on the HC-SR04 sensor. Because in the experiment presented here the distance d does not change, all we have to do is use the HC-SR04 sensor to measure the echo time t' for one given temperature T'_C, calculate the distance d for this temperature,

$$d = \frac{v't'}{2} = 20.05\sqrt{273.15 + T'_C}\,\frac{t'}{2}\,\text{m}, \qquad (4)$$

and then use this distance d in all the calculations (3) of the speed of sound at other temperatures. Equally well we could use an average of such distance measurements.

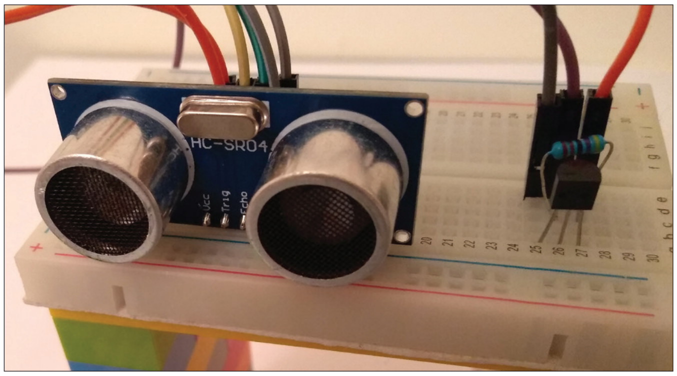

Fig. 1. The HC-SR04 and the DS18B20 sensors.

What can we say about the error in the measurement of the echo time t? Surprisingly, for the same distance d and at the same temperature T_C, the readings from the HC-SR04 sensor are not constant. Usually there is an echo time value that shows up more often, with some other values that show up less often. What echo time are we supposed to use for the calculation of the speed of sound? The easiest thing to do is to select just one echo time value,[2] presumably the one that shows up more often, and ignore the other echo time values. Another option is to calculate the average of these echo time values, for a given large number of echo time measurements. A third option is to calculate the speed of sound for each echo time measurement, and then calculate the average of these speed values.[1] Students are familiar with the process of calculating the average of a set of experimental values, with the goal of decreasing the random errors affecting the measurements, and the averaging performed in Ref. 1 seems reasonable. This averaging process is fully justified when we have a Gaussian distribution of errors, with the experimental values spread to the left and to the right of the true value. But do the measured echo time values always follow a Gaussian distribution?

The pattern followed by the measured echo time values emerged in full light when we reflected the

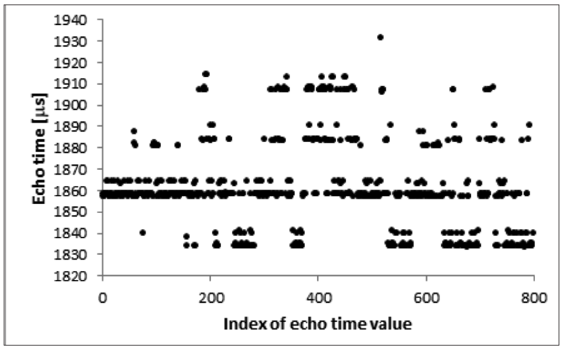

Fig. 2. A sample of 800 echo time values measured with a rectangular piece of cardboard placed at 30 cm in front of the HC-SR04 ultrasonic sensor. Arduino interrupts are enabled.

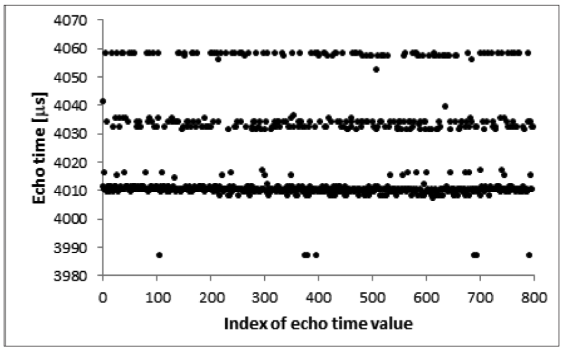

Fig. 3. A sample of 800 echo time values measured with a wall at 70 cm in front of the HC-SR04 ultrasonic sensor. Arduino interrupts are enabled.

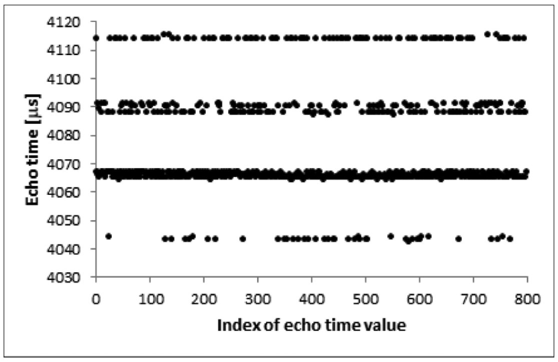

Fig. 4. A sample of 800 echo time values measured with a wall at 70 cm in front of the HC-SR04 ultrasonic sensor. Arduino interrupts are disabled.

burst of ultrasonic pulses on a rectangular piece of soft and not very flat cardboard (an envelope), a material on which the ultrasonic pulses did not reflect very well. The surface area of 38.0 cm × 26.5 cm = 1007 cm^2 was less than the 0.5 m^2 recommended in the HC-SR04 datasheet,[4] but this was compensated for by the fact that the piece of cardboard was at a relatively short distance of only 30 cm. As shown in Fig. 2, the echo time values belong to a set of quasi-discrete levels, resembling quantum energy levels.

A similar pattern was obtained when we increased the distance to 70 cm and we reflected the burst of ultrasonic pulses on a wall in front of the HC-SR04 sensor. The greater distance was compensated for by the large, flat, and hard surface of the wall. As shown in Fig. 3, the measured echo time values are again separated by gaps.

We obtained the most clear pattern when the 800 echo times were again measured with the wall at 70 cm in front of the HC-SR04 sensor, but this time with the Arduino interrupts disabled. As shown in Fig. 4, the spread of the measured values around the echo time levels is greatly reduced.

How can we understand this pattern? The key is to realize that the separation between the observed echo time levels is about 25 μs, which corresponds to a frequency of 40 kHz. This is exactly the ultrasonic frequency at which the HC-SR04 sensor emits the eight rectangular pulses. Although the electronic circuit of the HC-SR04 sensor is not described in its datasheet,[4] we can assume that the rectangular pulses have a cumulative effect. As a screen capture from a digital oscilloscope shows,[5] the voltage on the **echo** pin of the HC-SR04 sensor changes from 0 V to 5 V right after the burst of eight ultrasonic pulses is sent out. This voltage stays high while the ultrasonic pulses travel to the target, reflect off the target, and then return to the ultrasonic sensor. After a small number of these pulses are received back, the voltage on the **echo** pin of the HC-SR04 sensor changes from 5 V to 0 V, and this is how the echo time is determined. When the reflected pulses are received unattenuated, the ultrasonic sensor measures the correct echo time. What happens when some of the pulses are received attenuated, or missed altogether? In this case more pulses are needed, in order for the ultrasonic sensor to reach that threshold at which the **echo** pin voltage changes. If only one extra pulse is needed, then the reported echo time will be 25 μs longer. At 20 ^{o}C, this corresponds to an additional 4.3 mm in the calculated distance.

Unlike the special situations discussed above, for a large, flat, and hard surface (a wall) not too far away from the HC-SR04 sensor (at about 30 cm), the measured echo time values usually reveal only two echo time levels, or only one level. The averaging of all the echo time values is justified only in this latter case.

Other factors can affect the performance of the HC-SR04 sensor. If the sensor is placed directly on the floor, or on a desk, some ultrasonic pulses could scatter back because of the roughness of the horizontal surface. For this reason we recommend the mounting of the HC-SR04 sensor at some height above the floor or desk, as shown in Fig. 5.

Another issue is the size of the reflecting object. The HC-SR04 datasheet[4] mentions that, in order to achieve the best performance, the reflecting object must present a flat smooth surface of no less than 0.5 m^2. For this reason we recommend to use a wall. Smaller reflecting objects will result in longer echo time values, with corresponding shorter calculated speed of sound values.[2]

We have also noticed that the HC-SR04 sensor is very sensitive to air currents in the room. We cannot use fans in order to vary the temperature,[1] the air has to be completely still. For this reason we recommend to collect the experimental data in a room with closed windows and doors.

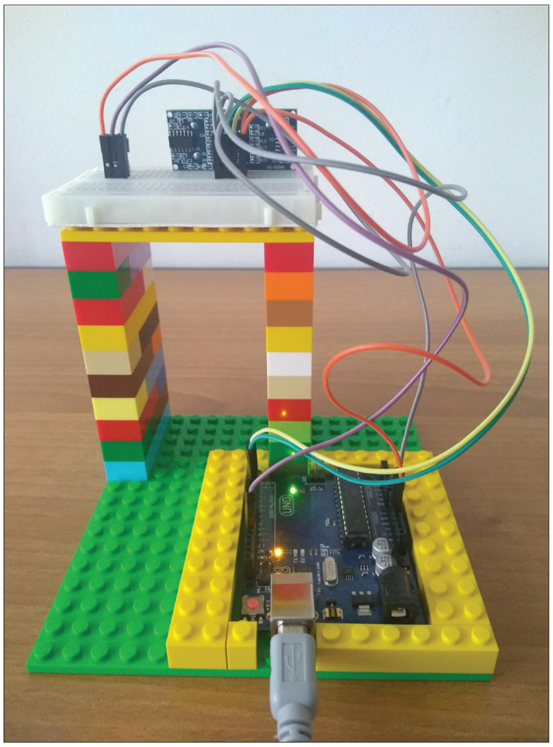

Fig. 5. The experimental setup.

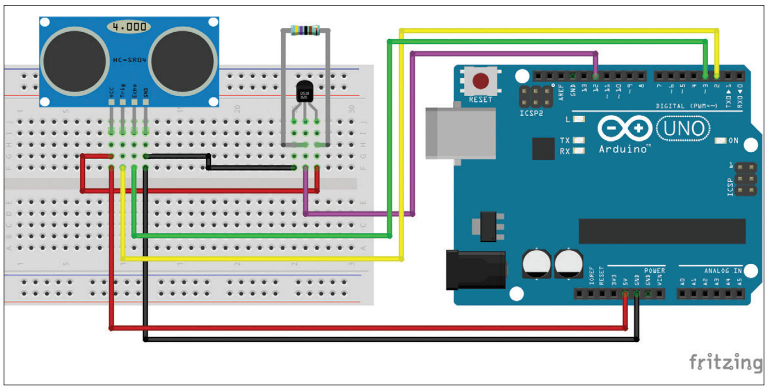

Fig. 6. The wiring of the electronic circuit.

The experimental errors related to the Arduino interrupts

In order to measure the echo time, an electronic device (Arduino Uno, digital oscilloscope, etc.) has to find out for how long the voltage on the **echo** pin of the HC-SR04 ultrasonic sensor stays high. The Arduino code relies on the **pulseIn()** function for this measurement. The **pulseIn()** function (implemented in the **wiring_pulse.c** file) calls the **countPulseASM()** function (implemented in the **wiring_pulse.S** file), which counts a number of CPU clock cycles. The **countPulseASM()** function, written in assembly language, runs a **while** loop during which a counter variable named **width** is continuously incremented for as long as the voltage on the specified Arduino pin stays high. In the Arduino 1.8.13 Integrated Development Environment the **wiring_pulse.c** and **wiring_pulse.S** source files are found in the **hardware\arduino\avr\cores\arduino** folder.

The fact that the Arduino interrupts are detrimental to the accuracy of the **pulseIn()** function is mentioned in the **wiring_pulse.c** file. During an Arduino interrupt the **width** variable stops from being incremented, and as a result the number of CPU clock cycles gets underreported. To avoid this behavior one must disable the Arduino interrupts, and this is done by placing the critical code in between **noInterrupts()** and **interrupts()** instructions.

By comparing the echo times from Figs. 3 and 4, we discover that, when the Arduino interrupts are enabled, two undesired things happen: the reported echo times are slightly shorter (1.3%) and slightly more spread around the echo time levels. This is because some Interrupt Service Routines are serviced during all of the echo time measurements (timer interrupts fit this behavior) and some Interrupt Service Routines are serviced only during some of the echo time measurements.

This being said, one should not rush to disable the Arduino interrupts everywhere. In order for the **micros()**, **millis()**, and **delay()** Arduino functions to work properly, the timer interrupts must be enabled.

The experimental investigation of the speed of sound vs. temperature

For the actual experiment presented here we have turned off the heat during winter, and we have allowed the room temperature to slowly decrease from 25.31 ^{o}C to 10.31 ^{o}C during three cold, cloudy days. We recommend to do this investigation in winter not only because of the large temperature variation that can be obtained in this way, but also because the speed of sound depends on humidity, and Eq. (1) applies only to dry air. When the outside temperature is below the freezing point, the air inside is mostly dry.

The experimental setup, shown in Figs. 1 and 5, consists of an Arduino Uno microcontroller board, an HC-SR04 ultrasonic distance sensor, and a DS18B20 temperature sensor together with its 4.7 kΩ pull-up resistor. The breadboard with the electronic components is mounted up high on some Lego bricks, at a distance of 30 cm in front of a wall. Keep in mind that for an accurate temperature measurement the black DS18B20 sensor must be kept out of direct sunlight.

As shown in Fig. 6, the **GND** pins of the sensors are connected to an Arduino **GND** pin, and the **VCC** pins of the sensors are connected to the Arduino **5 V** pin. The **trig** pin of the ultrasonic sensor is connected to **digital pin 2**, and the **echo** pin is connected to **digital pin 3**. The **data** pin of the temperature sensor is connected to **digital pin 12.** A detailed description of the HC-SR04 sensor, and the Arduino code that measures the echo time, are given in Ref. 6. A detailed description of the DS18B20 sensor and the Arduino code that measures the temperature are given in Ref. 7. The DS18B20 temperature sensor has an accuracy of ±0.5 ^{o}C.[8]

The Arduino program[9] used in our experimental investigation has two parts. In the first part, the **setup()** function, the resolution of the temperature sensor is set to 12 bits. In the second part, the **loop()** function, the experimental data is collected and partially analyzed. About every two minutes, one temperature measurement and 800 echo time measurements are taken. The average of all 800 echo time values and

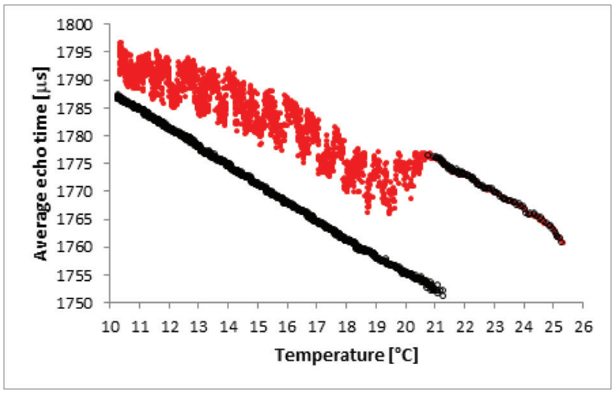

Fig. 7. The average echo time as a function of temperature. We compare the average of all 800 echo time values (solid red circles) with that of the short echo time values (open black circles).

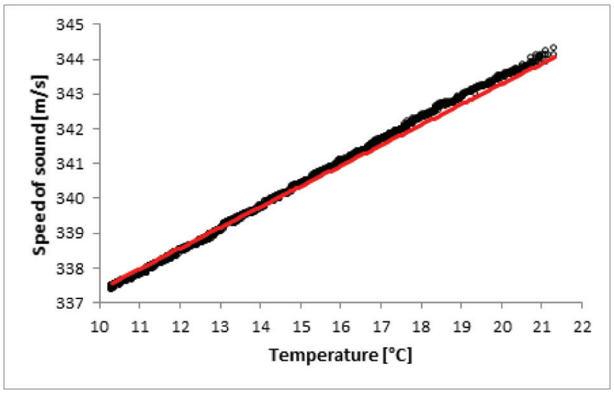

Fig. 8. The experimentally determined speed of sound as a function of temperature (black circles) matches very well the theoretical formula (red line).

the average of the short echo time values are calculated. The short echo times are selected by the condition: echo time—shortest echo time < 12 μs. For each measured temperature T_C the program also calculates the distance d_{short} traveled by the ultrasonic wave, as derived from Eqs. (3) and (1):

$$d_{\text{short}} = \frac{vt_{\text{short}}}{2} = 20.05\sqrt{273.15 + T_C}\,\frac{t_{\text{short}}}{2}\ \text{m}\,, \tag{5}$$

where t_{short} is the average of the short echo time values. These calculated distances d_{short} are needed during the subsequent data analysis.

In Fig. 7 we show the average echo time as a function of temperature. The solid red circles represent the average of all 800 echo time values, while the open black circles represent the average of the short echo time values. We realize that calculating the average of all 800 echo time values can be misleading. In the 10 to 21 $^{\text{o}}$C temperature range, the red circles go up and down according to the random ways in which the echo time levels are populated. On the other hand, the black circles display a consistent pattern. There are two categories of black circles that we see in the graph. We have the data points for which no ultrasonic pulses are lost; these are the black circles below, with a reported d_{short} distance of about 30.1 cm. And we have the data points for which one ultrasonic pulse is lost; these are the black circles above, with a reported d_{short} distance of about 30.5 cm. The red and the black circles overlap in the 21 to 25 $^{\text{o}}$C temperature range where all, or almost all, of the 800 measured echo times belong to the same echo time level. Unfortunately, these are exactly the data points that have to be discarded, due to the lost ultrasonic pulses.

We select only the data points for which no ultrasonic pulses are lost in at least one of the 800 echo time measurements, the data points with a reported d_{short} distance of about 30.1 cm. For this restricted set of data points, the average of the reported d_{short} distances is 30.144 cm. Next, using the average of the short echo time values as the time t, and the average distance of 30.144 cm as the distance d, we calculate the speed of sound according to Eq. (3). In Fig. 8 we show the variation of this measured speed of sound with temperature, together with the theoretical curve given by Eq. (1). The agreement between the experimentally determined values and the theoretical prediction is excellent.

The calibration procedure that gave us the distance [Eq. (4)] also helps us understand why the experimental error with which the DS18B20 sensor measures the temperature does not affect the excellent match seen in Fig. 8. The experimental speeds are given by

$$v = \frac{2d}{t} = 20.05\sqrt{273.15 + T'_C}\,\frac{t'}{t}\,\text{m/s} \approx (331.4 + 0.6\,T'_C)\frac{t'}{t}\,\text{m/s}\,, \tag{6}$$

while the theoretical speeds are given by

$$v = 20.05\sqrt{273.15 + T_C}\ \text{m/s} \approx (331.4 + 0.6\,T_C)\ \text{m/s}\,. \tag{7}$$

The ratio t'/t is very close to one, accounting for the less than 2% relative variation of the speed of sound. A small offset ΔT in the measurement of the T'_C and T_C temperatures will vertically shift both Eqs. (6) and (7) on the graph with approximately the same amount. This argument holds true not only when the distance d is the outcome of one measurement, but also when it is the average of several measurements made at different temperatures.

Looking at Eq. (6) we also notice that, when both t' and t are reduced by 1.3%, the ratio t'/t does not change. This explains why we can perform this speed of sound investigation with the Arduino interrupts enabled, and still obtain good results.

Conclusions

The Arduino investigation of the speed of sound presented here serves several purposes. While the main goal is to verify experimentally the theoretical formula in Eq. (1), we also use this integrated STEM activity to teach students about electronics and computer programming. Especially important is the idea that one should not treat an electronic sensor as a black box, but instead try to understand its inner workings. In particular, the echo time values reported by the HC-SR04 sensor do not display a Gaussian distribution, and as a result we simply cannot average all of these values in order to decrease the random errors. What is needed is the average of the short echo time values, those on the lowest echo time level.

The author is very much indebted to the three anonymous reviewers for their very valuable comments. The manuscript has greatly benefitted from their feedback. One of the reviewers mentioned the need to disable the Arduino interrupts, a very important implementation detail.

References

1. Marcelo Dumas Hahn, Frederico Alan de Oliveira Cruz, and Paulo Simeão Carvalho, "Determining the speed of sound as a function of temperature Using Arduino," *Phys. Teach.* **57**, 114–115 (Feb. 2019).
2. M. Oprea and Cristina Miron, "Didactic experiments for determining the speed of sound in the air," *Romanian Reports in Physics* **68** (4), 1621–1640 (2016).
3. Sound Speed in Gases, http://hyperphysics.phy-astr.gsu.edu/hbase/Sound/souspe3.html.
4. HC-SR04 Datasheet, https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf.
5. PC Services, Ultrasonic Distance Sensing Using HC-SR04, http://www.pcserviceselectronics.co.uk/arduino/Ultrasonic/.
6. Calin Galeriu, Scott Edwards, and Geoffrey Esper, "An Arduino investigation of simple harmonic motion," *Phys. Teach.* **52**, 157–159 (March 2014).
7. Calin Galeriu, "An Arduino investigation of Newton's law of cooling," *Phys. Teach.* **56**, 618–620 (Dec. 2018).
8. DS18B20 Datasheet, https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf.
9. The Arduino program echo_time_temp.ino is provided as a supplemental online material and can be found at *TPT Online*, http://dx.doi.org/10.1119/10.0009993, under the Supplemental tab.

Calin Galeriu *teaches physics and chemistry at Mark Twain International School. He has a BS degree in physics from the University of Bucharest, an MA degree from Clark University, and a PhD degree from Worcester Polytechnic Institute.*
Mark Twain International School, Strada Erou Iancu Nicolae 89-93, Voluntari, IF, Romania; calin.galeriu@marktwainschool.ro