RTSS-MS

Real Time Speed of Sound – Measuring Setup

Parth Kadam
Department of Electronics and
Telecommunication Engineering,
Sardar Patel Institute of Technology
(Affiliated to University of Mumbai)
Andheri (West), Mumbai
parth.kadam24@spit.ac.in

***Abstract*—RTSS-MS is setup that measures the speed of sound in real time by combining ultrasonic time-of-flight data with environmental parameters – Temperature and Humidity. The setup uses an ESP32 microcontroller to acquire sensor data, calculate both theoretical and experimental speed of sound values, and log validated results to a cloud platform. The system is designed for real-time analysis, model validation, and detection of experimental inconsistencies.**

Keywords—speed of sound, embedded systems, IoT, Wong-Embleton model, real-time monitoring, acoustic measurement

Introduction

The speed of sound in air is influenced by temperature, humidity, and pressure, making its accurate measurement a complex task in acoustic physics. While theoretical and empirical models exist to estimate sound speed, real-time validation using live sensor data remains uncommon in embedded systems. RTSS-MS addresses this gap by enabling real-time acquisition, computation, and cloud-based logging of both theoretical and experimental values of sound speed.

This work uses the Wong–Embleton model, which refines the dependence of sound speed on the two easily available environmental parameters – Temperature and Humidity. RTSS-MS also includes a logic function to reject misaligned readings by comparing the two speed values with a strict threshold.

.

I. Theory

A. The Wong-Embleton Empirical Scale[1]

The Wong–Embleton formulation accounts for the variation of the speed of sound ccc in humid air as a function of relative humidity H and temperature T. Their key result provides:

$$\frac{c_0}{c_h} = 1 + \frac{H}{100}\left(9.66 \times 10^{-4} + 7.2 \times 10^{-5} T + 1.8 \times 10^{-6} T^2 + 7.2 \times 10^{-8} T^3 + 6.5 \times 10^{-9} T^4\right)$$

where c_0 is the speed in dry air, and c_h is the speed in humid air. H is measured in %age relative humidity and T is measured in °C. RTSS-MS uses this formula to compute S(T, H), ie. speed of sound as a function of Temperature and Humidity based on real-time readings from a DHT11 sensor.

II. System Overview

A. Hardware

- ESP32 (DOIT DevKit V1) - MCU with a built-in Wi-Fi module.
- DHT11 – Digital Humidity (±5%) & Temperature (±2°C) sensor[2].
- HC-SR04 - Ultrasonic module to measure time-of-flight (range 2–400 cm)
- Breadboard, jumper wires, USB power
- Omega set-squares to keep the setup well-aligned

B. Software & Logic

- ISR (Interrupt Service Routine) on the ESP32 captures rising/falling edges of the ultrasonic echo pin to calculate time Δt.

```
void IRAM_ATTR timeRec(){
  if (digitalRead(echo) == HIGH) {
    start = esp_timer_get_time();
  } else {
    end = esp_timer_get_time();
    echo_received = true;
  }
}
```

  Accuracy up to nanoseconds can be achieved using the ESP's `esp_timer_get_time()` function[3].

- Distance D is pre-measured using appropriate tools (set squares, in this case 25.6cm)
- Experimental speed S(D, Δt) = D / Δt
- Theoretical speed S(T, H) is calculated using the Wong–Embleton equation
- alignmentCheck() uploads data to ThingSpeak only if |S(D, Δt) – S(T, H)| < 10m/s, since the error due to misalignment or DHT inaccuracy is detectable.

III. Error Analysis

The setup is arranged in this way such that the Ultrasonic sensor is some fixed distance away from a reflecting surface. One has to make sure if the sensor is aligned correctly in this setup. Mere misalignment of a few degrees can introduce significant error. However to avoid logging misaligned readings, alignmentCheck() comes into play. On experimental basis it is confirmed that in this particular setup, the error/difference is exponential, ie. either the error %age is <0.2% (off by ~0.05m/s) or >7% (off by 30m/s) – there is no between. And this can shoot up to 60% on subtle carelessness. Besides, DHT11 can give an error up to even 9-10m/s due to ±2°C and ±5% accuracy. Hence, in this particular setup, we used the threshold of 10m/s. Anything off than this won't be uploaded to ThingSpeak. It must be noted that data collection showcased in this report was done in a properly aligned setup

on a monsoon day, such that alignmentCheck() never came into the play.

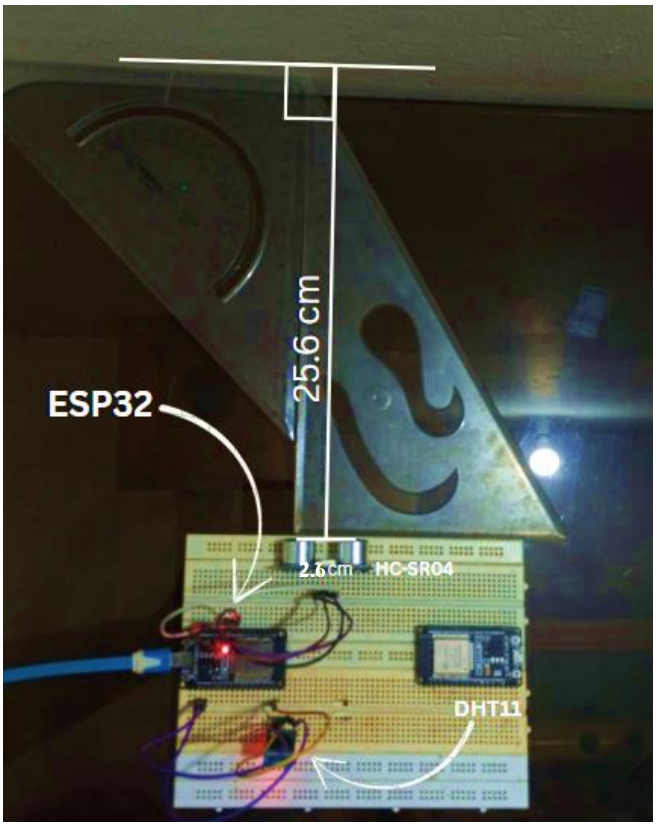
Other than this, safety code snippets have been implemented if either of the sensors are not connected properly.

```
if(isnan(X)) return -1;
```

Where X can be T or H.

Despite all this, there are some errors which can't be dealt with int the current setup and hardware - DHT11 is known for its non-reliable nature as it is primarily made for DIY hobby projects. Instead, one can use DHT22 or SHT31 for more reliable T and H readings. Besides, ESP's ISR jittering can give minor inaccuracies which can be tackled by taking multiple readings for verification purpose, and using an industry grade dev board like STM32 which isn't fool-proof either but more reliable in this case.

IV. The Setup



A. Setting Up Things

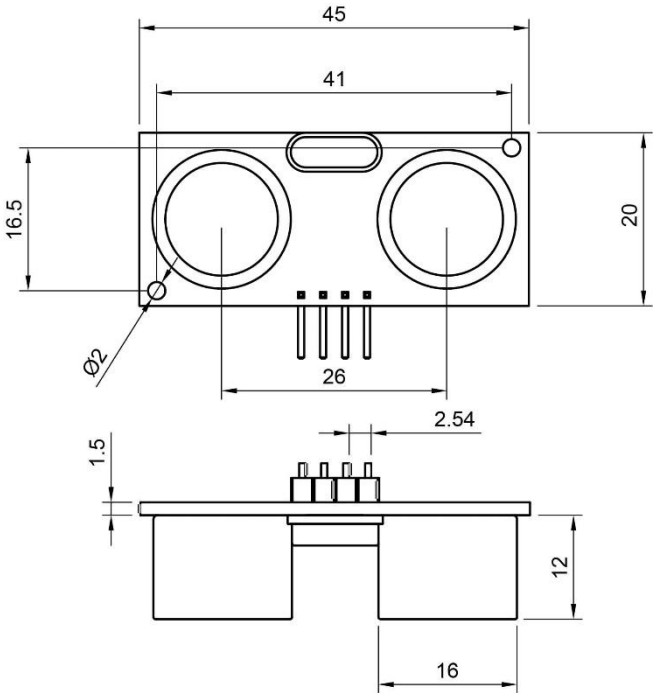
The setup should be well aligned according to your convenience. Here, Omega set-squares have been used for alignment convenience.

1) First, measure the perpendicular distance from the reflecting surface: In this case, it is a flat wall. Input the distance in cm in the RTSS-MS.ino file's line 6.

```
#define DISTANCE_CM 25.6
```

2) Account for offsets: We have to take the following things into consideration:

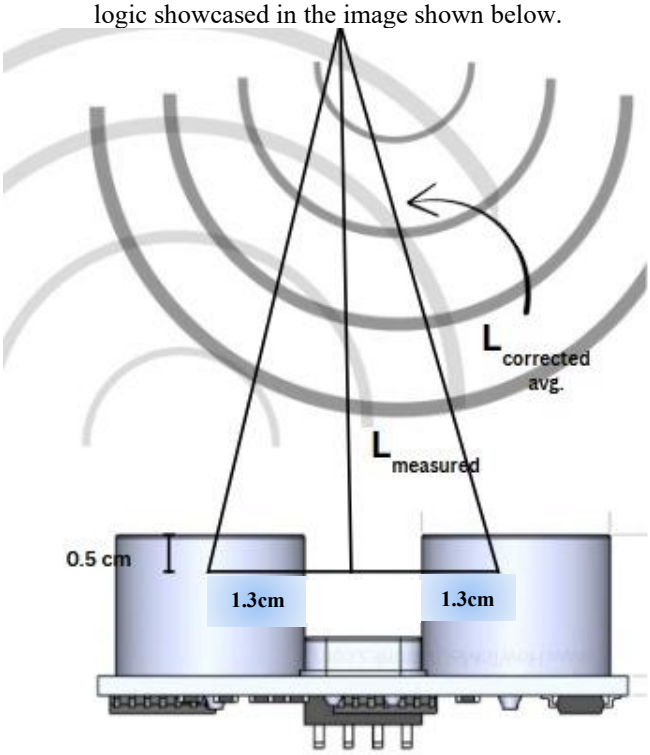
a) Distance betweem transmitter and receiver[4]*:* The distance between transmitter and receiver in HC-SR04 is ~2.6cm



b) Geometrical correction: The calculations.cpp file takes care of this offset by correcting the distance travelled using pythagoras formula:

```
float D_corrected = sqrt(sq(D) + sq(1.3))/100;
```

An offset of 1.3 (half of 2.6) is considered based on the logic showcased in the image shown below.



c) Hollowness of the transmitter and receiver: The Transmitter and Receiver cylinders are 0.5cm deep before the actual hardware starts. We have to account for this distance as well while considering `#define DISTANCE_CM`.

B. *Running the program*

The Program is well-documented in RTSS-Monitoring-Setup Repository on GitHub[5]. RTSSv0.7 was used as a debug mode, and used a simplified empirical formula. RTSS-MS (v1) is modular and uses Wong-Embleton Empirical Scale instead, which increases the accuracy well.

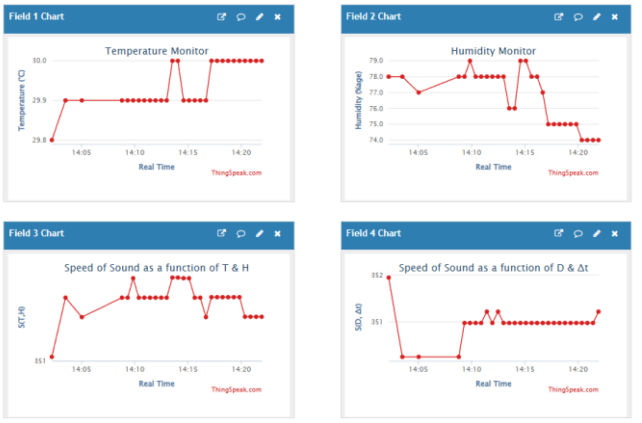
1. First download all the files, make sure all of them are in a folder named RTSS-MS, or that the .ino file and the folder are named same.
2. Edit GPIO no.s and distance D according to your setup. Fill in your Wi-Fi and ThingSpeak credentials properly. More about ThingSpeak later.
3. Check alignment, you can use v0.7 for this purpose.
4. Flash the program into the MCU. These particular files are compactible with Arduino IDE, one can use ESP-IDF as well if comfortable with .cpp files.

V. Data Acquisition

A. *ThingSpeak data logging*

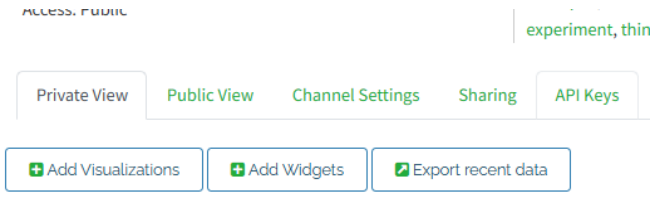
ThingSpeak is an IoT platform to upload data and work on real time analysis. In this particular setup, ThingSpeak can log T, H, S(T, H) and S(D, Δt)). In order to do this, log in to ThingSpeak and create a channel. Label the fields as mentioned in the above mentioned order. Before flashing the prgram into the MCU, make sure you put `channelID` and `writeAPIkey` carefully in WiFiconfig.h, else data won't be logged. While the setup is running, it will push the data to ThingSpeak evey 15 seconds.

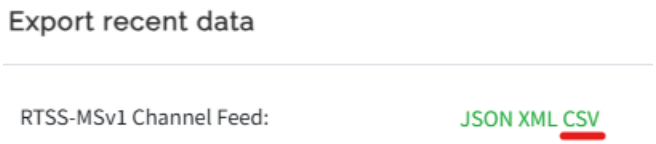
Readings on 22 June 2025[6] (A fair monsoon day in MMR)



B. *Data in tabular form*

In the private view of your channel, click on Export Recent Data and then click on CSV hyperlink of your channel's feed data





You will get a .csv file which you can later convert to .xlsx on excel to proceed further with your data.

On 22 June 2025, a total of 29 readings were taken using RTSS-MS:

TABLE I. 22 June 2025 Readings

S. No.	*Date & Time of Entry (UTC)*	*T (°C)*	*H (% RH)*	*S(T,H) (ms^{-1})*	*S(D,Δt) (ms^{-1})*
1	*2025-06-22 08:32:15*	*29.8*	*78*	*351.17548*	*351.94403*
2	*2025-06-22 08:33:32*	*29.9*	*78*	*351.24673*	*350.26007*
3	*2025-06-22 08:35:04*	*29.9*	*77*	*351.22330*	*350.26007*
4	*2025-06-22 08:38:50*	*29.9*	*78*	*351.24673*	*350.26007*
5	*2025-06-22 08:39:22*	*29.9*	*78*	*351.24673*	*350.97980*
6	*2025-06-22 08:39:53*	*29.9*	*79*	*351.27014*	*350.97980*
7	*2025-06-22 08:40:25*	*29.9*	*78*	*351.24673*	*350.97980*
8	*2025-06-22 08:40:56*	*29.9*	*78*	*351.24673*	*350.97980*
9	*2025-06-22 08:41:28*	*29.9*	*78*	*351.24673*	*351.22034*
10	*2025-06-22 08:41:59*	*29.9*	*78*	*351.24673*	*350.97980*
11	*2025-06-22 08:42:31*	*29.9*	*78*	*351.24673*	*351.22034*
12	*2025-06-22 08:43:02*	*29.9*	*78*	*351.24673*	*350.97980*
13	*2025-06-22 08:43:34*	*30.0*	*76*	*351.27090*	*350.97980*
14	*2025-06-22 08:44:05*	*30.0*	*76*	*351.27090*	*350.97980*
15	*2025-06-22 08:44:36*	*29.9*	*79*	*351.27014*	*350.97980*
16	*2025-06-22 08:45:08*	*29.9*	*78*	*351.24673*	*350.97980*
17	*2025-06-22 08:45:40*	*29.9*	*78*	*351.24673*	*350.97980*
18	*2025-06-22 08:46:11*	*29.9*	*78*	*351.24673*	*350.97980*
19	*2025-06-22 08:46:43*	*29.9*	*77*	*351.22330*	*350.97980*
20	*2025-06-22 08:47:14*	*30.0*	*75*	*351.24731*	*350.97980*
21	*2025-06-22 08:47:46*	*30.0*	*75*	*351.24731*	*350.97980*
22	*2025-06-22 08:48:17*	*30.0*	*75*	*351.24731*	*350.97980*
23	*2025-06-22 08:48:48*	*30.0*	*75*	*351.24731*	*350.97980*
24	*2025-06-22 08:49:20*	*30.0*	*74*	*351.22372*	*350.97980*
25	*2025-06-22 08:49:52*	*30.0*	*74*	*351.22372*	*350.97980*
26	*2025-06-22 08:50:23*	*30.0*	*74*	*351.22372*	*350.97980*
27	*2025-06-22 08:50:54*	*30.0*	*74*	*351.22372*	*350.97980*
28	*2025-06-22 08:51:26*	*30.0*	*74*	*351.22372*	*350.97980*
29	*2025-06-22 08:51:57*	*30.0*	*74*	*351.22372*	*351.22034*

[a] Readings in .xlsx format available on GitHub Repository of RTSS-MS

The readings suggest that both Wong-Embleton Formula and RTSS-MS complement each other, with a max difference

of 0.98666 m/s between the two and an average difference of 0.28288 m/s, which is acceptable, considering ±5% RH, ±2°C of DHT11, lack of local variables in ISR function in .ino file and human errors.

CONCLUSION

A total 29 valid readings taken on a monsoon day in Mumbai Metropolitan Region. The observed averages were:

$S(T, H)_{average} = 351.2437\ ms^{-1}$

$S(D, \Delta t)_{average} = 350.9653\ ms^{-1}$

Which shows a mean deviation of merely 0.2784 m/s - well within the expected error range of the sensors used. These results demonstrate a strong agreement between the Wong-Embleton Empirical Scale Formula and the RTSS-MS method, suggesting that both models can reliably complement each other in real-time environmental measurements.

The consistency across datasets also validates the operational accuracy of the prototype under humid monsoon conditions, which typically introduce greater variability in sensor readings. Future work may include hardware improvements for RTSSv2, such as migration to a more efficient microcontroller platform like the STM32 using bare-metal ISR (Interrupt Service Routine) implementations to enhance timing precision. Additionally, replacing the current temperature-humidity module with a DHT22 or a higher-grade sensor could yield even more stable and accurate measurements across a wider range of atmospheric conditions

ACKNOWLEDGMENT

This work was conceived, developed, and executed independently by the author, without the use of institutional resources or external funding.

REFERENCES

[1] G. S. K. Wong and T. F. W. Embleton, "Variation of the speed of sound in air with humidity and temperature," *J. Acoust. Soc. Am.*, vol. 77, no. 5, pp. 1710–1712, May 1985.

[2] Mouser Electronics, "DHT11 Temperature and Humidity Sensor Datasheet," [Online]. Available: https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf

[3] Espressif Systems, "ESP32 Technical Reference Manual: pinout, timers and peripherals," [Online]. Available: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf

[4] Makerguides.com. "HC-SR04 Ultrasonic Sensor with Arduino Tutorial (5 Examples)." *Makerguides*, published ~6.4 years ago. Available online with CAD drawings and dimension diagrams (credited under CC BY-NC-SA 4.0) : https://www.makerguides.com/hc-sr04-arduino-tutorial/

[5] Author's GitHub repository by username kar-Madhu, "RTSS-Monitoring-Setup" Available: *https://github.com/kar-Madhu/RTSS-Monitoring-Setup/*