AI & SUSTAINABILITY IN VET EDUCATION

Name of the Author/s: Sucessos Criativos, Lda

Title of WP: WP3 – AI-SAVES@ProjectHub

Activity Reference: 3.2 – Project template for AI-SAVES@ProjectHub

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Table of Contents

1. Project Title: 4

2. Project Category: 4

3. Project Description: 4

4. Keywords: 4

5. Target Audience: 4

6. Project Objectives: 5

7. Project Timeline: 5

8. Resources Provided: 5

9. Tools and Technologies Used: 6

10. Step-by-Step Implementation Plan: 6

Step 1: Prepare the Raspberry Pi for Hardware Interaction 6

Step 2: Connect the Ultrasonic Sensor 7

Step 3: Write the Python Program to Measure Distance 8

Step 4: Run the Program and Observe The Output 10

11. Evaluation Criteria: 11

12. Languages Available: 12

13. Contributors: 12

17. Feedback and Continuous Improvement: 12

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

1. Project Title:

First Steps with AI-VET Kit: Measuring Distance with an Ultrasonic Sensor

---

2. Project Category:

Microelectronics Fundamentals

---

3. Project Description:

This project introduces learners to distance measurement using the HC-SR04 ultrasonic sensor. Participants will connect the sensor to a Raspberry Pi 5 and develop a Python script to measure and display distance values in real-time. This hands-on activity is a basis for automation and IoT applications such as smart bins, robotics, or energy-efficient systems.

---

4. Keywords:

Raspberry Pi, Python, HC-SR04, GPIO, Distance Sensor, Automation, Environmental Sensing

---

5. Target Audience:

Educators, learners, and makers with or without prior experience who are beginning their journey into microelectronics and physical computing.

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

6. Project Objectives:

- Connect and configure the HC-SR04 ultrasonic sensor with a Raspberry Pi 5
- Measure real-world distance using digital signal processing in Python
- Reinforce skills in GPIO access, coding, and sensor-based input/output

---

7. Project Timeline:

The estimated duration is approximately **2.5 to 3 hours**, including environment setup and physical assembly.

---

8. Resources Provided:

- AI-VET Kit (Raspberry Pi 5, LED, Breadboard, Jumpers, HC-SR04 Ultrasonic Sensor)

MATERIAL	
Raspberry Pi 5	
27W USB Type-C Power Supply for Raspberry Pi 5	
Flat Jumper Wires F/F	

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Breadboard	
SINGLE-FILE WIRE FOR BREADBOARDS	
HC-SR04P - ULTRASONIC SENSOR 3V~5V	

---

9. Tools and Technologies Used:

- Raspberry Pi OS
- Python 3 (in virtual environment)
- `lgpio` library
- Geany IDE
- Terminal (bash)

---

10. Step-by-Step Implementation Plan:

Step 1: Prepare the Raspberry Pi for Hardware Interaction

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

This project builds directly on the tasks completed in “Getting Started with Raspberry Pi 5: Setup, Configuration, and Hello World in Python.” and “Blinking an LED”. If you're starting here, or need a refresher, you’ll first go through the essential setup steps to prepare your Raspberry Pi to control physical components.

In this step, you will:

- Install Raspberry Pi OS using the Raspberry Pi Imager tool.
- Configure basic system settings such as Wi-Fi, keyboard layout, and time zone.
- Enable key interfaces including SPI, I2C, 1-Wire, and VNC using the Raspberry Pi Configuration menu.
- Identify the Raspberry Pi’s IP address using ifconfig and connect remotely via TigerVNC Viewer.
- Perform a full system update
- Install the lgpio library

Step 2: Connect the Ultrasonic Sensor

In this step, you'll wire the HC-SR04 ultrasonic sensor to the Raspberry Pi. The sensor works by sending an ultrasonic pulse from the **TRIG** (trigger) pin and waiting for it to bounce back to the **ECHO** (echo) pin. The time it takes for the echo to return is used to calculate the distance to an object.

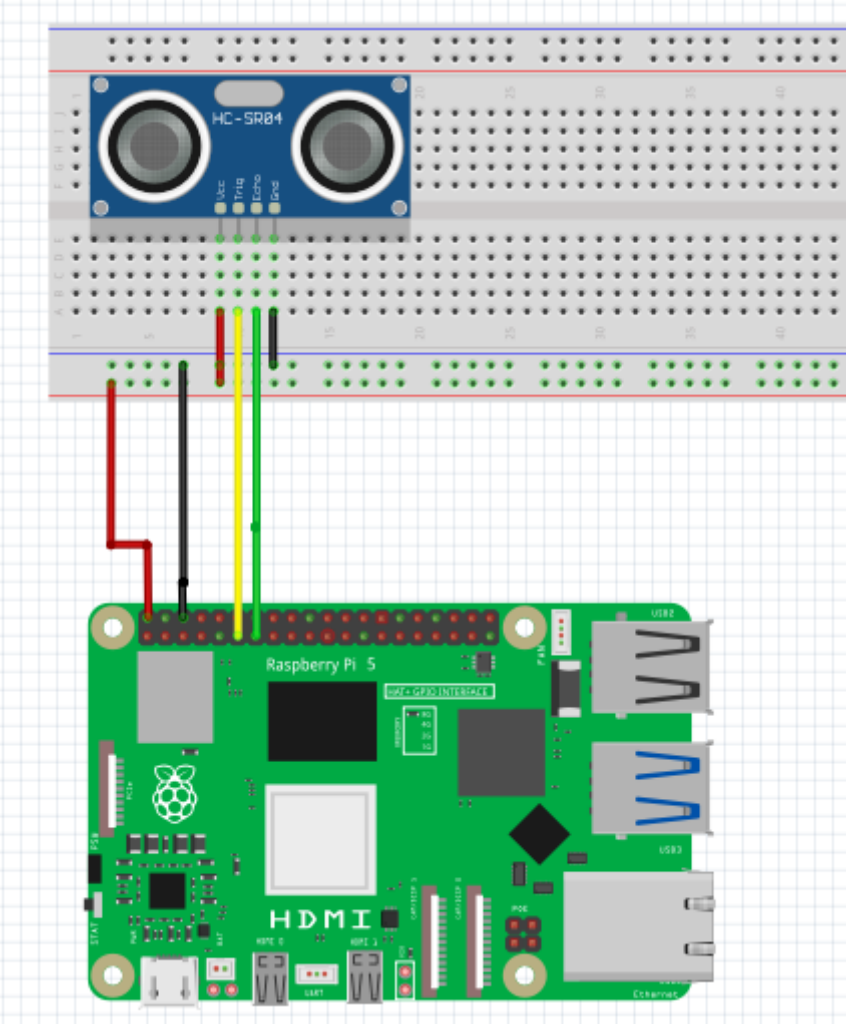
- **TRIG**: Sends a high-frequency sound pulse (trigger signal) when activated.
- **ECHO**: Receives the reflected pulse; the time gap between TRIG and ECHO helps determine distance.

Connections:

- **VCC** → Raspberry Pi 5V (pin 2)
- **GND** → Raspberry Pi GND (pin 6)
- **TRIG** → GPIO 17 (pin 11)
- **ECHO** → GPIO 27 (pin 13)

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



Step 3: Write the Python Program to Measure Distance

With the sensor connected, you'll now write a Python program that checks how far the object is, and prints a message in the terminal based on the result.

Open Geany and enter the following code:

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
import lgpio
import time

# Define GPIO pins for the ultrasonic sensor
TRIGGER_PIN = 17  # GPIO pin connected to the Trigger pin of the ultrasonic
sensor
ECHO_PIN = 27     # GPIO pin connected to the Echo pin of the ultrasonic
sensor

# Initialize GPIO chip (the default GPIO chip is "/dev/gpiochip0")
h = lgpio.gpiochip_open(0)

# Set the trigger pin as output and echo pin as input
lgpio.gpio_claim_output(h, TRIGGER_PIN)
lgpio.gpio_claim_input(h, ECHO_PIN)

# Function to measure distance using the ultrasonic sensor
def measure_distance():
    """
    Sends a pulse to the ultrasonic sensor and calculates the distance
    based on the duration of the returned echo.
    """
    # Send a 10 microsecond pulse to trigger the sensor
    lgpio.gpio_write(h, TRIGGER_PIN, 1)
    time.sleep(0.00001)  # Wait for 10 microseconds
    lgpio.gpio_write(h, TRIGGER_PIN, 0)

    # Wait for the echo to go high (start of the echo pulse)
    start_time = time.time()
    while lgpio.gpio_read(h, ECHO_PIN) == 0:
        start_time = time.time()  # Capture the start time when the pulse is
detected

    # Wait for the echo to go low (end of the echo pulse)
    end_time = time.time()
    while lgpio.gpio_read(h, ECHO_PIN) == 1:
        end_time = time.time()  # Capture the end time when the pulse ends

    # Calculate the pulse duration
    pulse_duration = end_time - start_time

    # Calculate the distance in centimeters (speed of sound is 34300 cm/s)
    # We divide by 2 because the sound travels to the object and back
    distance = (pulse_duration * 34300) / 2

    return distance

try:
    while True:
        # Measure the distance using the ultrasonic sensor
        distance_cm = measure_distance()
        print(f'Distance: {distance_cm:.2f} cm')  # Print the distance in
centimeters

        # Wait for 0.3 seconds before the next measurement
        time.sleep(0.3)

except KeyboardInterrupt:
    # Handle user interruption (Ctrl + C) to stop the program
```

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
    # Clean up by closing the GPIO chip
    lgpio.gpiochip_close(h)
    print("GPIO cleaned up!")
```

Save the file as **UltraSound.py.**

Code Breakdown:

Setup and Imports:The script starts by importing lgpio to manage GPIO pins and time to control timing and delays between sensor readings.

Pin Definition and Initialization:Two GPIO pins are defined: GPIO 17 for the TRIG pin and GPIO 27 for the ECHO pin. These are then configured using lgpio.gpio_claim_output() and lgpio.gpio_claim_input() respectively after opening the GPIO chip.

Measurement Function:The measure_distance() function triggers the ultrasonic sensor by sending a 10-microsecond pulse. It then waits to receive the returning echo. The time between sending and receiving the signal is used to calculate distance.

If the echo is received quickly → the object is close

If it takes longer → the object is farther away

The formula used is:

distance = (duration × speed of sound) / 2

Where the speed of sound is 34300 cm/s.

Main Loop: The program continuously measures and prints the distance every 0.3 seconds. This makes it responsive to real-time changes in object proximity.

Interrupt Handling: When the user presses Ctrl + C, the loop stops gracefully and the GPIO chip is safely closed using lgpio.gpiochip_close().

Step 4: Run the Program and Observe The Output

With the code written and saved, it's time to test the setup and observe how it responds to nearby objects.

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

To run the script:

Save the file as UltraSound.py

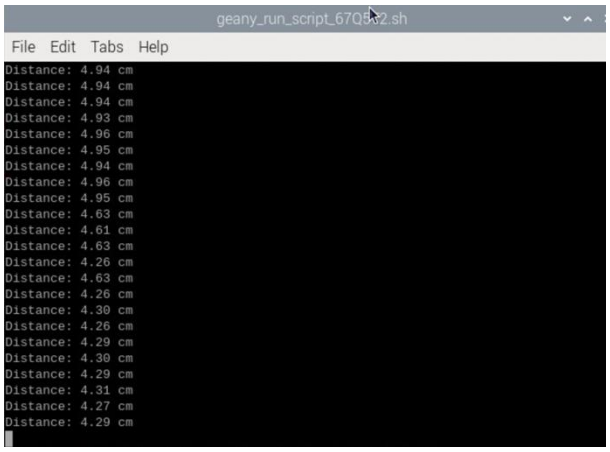
Press F5 in Geany to **execute** the code, or **run** it via the **terminal** by **typing**:

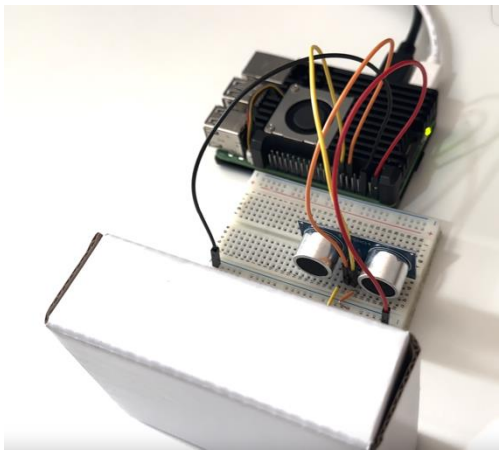
python3 UltraSound.py

As the script runs, it will measure and print the distance between the sensor and any object placed in front of it, updating every 0.3 seconds.

Object in front of sensor → prints smaller distance

Object far away → prints larger value





---

11. Evaluation Criteria:

-
- **Sensor Connection**: Ultrasonic sensor is properly wired and functional.
- **Digital Logic**: The script successfully detects object proximity based on echo timing.

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

- **Code Functionality**: No errors; the script runs smoothly and exits cleanly.
- **Hands-On Testing**: User tested the sensor with objects at varying distances and observed expected results.

---

12. Languages Available:

English, **Portuguese** (translations in progress)

---

13. Contributors:

Fátima Pais, Luís Soares – Project Designers and Facilitators

---

14. Feedback and Continuous Improvement:

- **Surveys**: At the end of this milestone, students and educators will complete feedback surveys to assess the effectiveness of the project.
- **Project Reviews**: After project completion, a final review meeting will be held to discuss potential improvements for future AI-based projects.
- **Data Analysis**: Feedback data will be analyzed to identify strengths and areas for improvement, informing future projects.

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.