Voice Language Action Framework For Electronic Sensor Control Systems

Adam Kacher[1], Fabio Osorio[2] and Robinson Jiménez-Moreno[3]
[1]CPE Lyon, Campus Lyontech, 3 rue Victor Grignard, Villeurbanne, France, adam.kacher@cpe.fr
[3]Department of Mecatronica, Universidad Militar Nueva Granada, Bogotá, Colombia,{est.fabio.osorio, robinson.jimenez}@unimilitar.edu.co

Abstract*– This document presents a human–robot interaction system that continuously measures the distance between an ultrasonic sensor and objects that represent obstacles. Additionally, it uses voice commands to adjust the distance thresholds that trigger the system's visual alerts, and it is integrated into an STM32 platform. The proposed architecture decouples computationally intensive AI processing from embedded control, running the audio capture workflow, local transcription using Whisper, and intent extraction using the Qwen 3.5 model on a computer, while the STM32 board is responsible exclusively for measuring distance and generating alert signals on an LED board, resulting in a more natural, intuitive, and flexible robotic control interface. The experimental results, obtained using 41 historical commands and various operators, show average processing times ranging from tens to just over a hundred seconds and a consumption of around a thousand tokens per instruction. The results support the feasibility of using large language models running locally as a semantic bridge between a user's voice and the control of physical parameters in robotics, thereby opening a line of research toward more human-like multimodal interfaces for autonomous systems and embedded platforms.

Keywords*-- Whisper, Voice control, LLM, Qwen 3.5, Ultrasonic sensor.

I. Introduction

Among the recent advances in artificial intelligence (AI), long language models (LLMs) have revolutionized many fields, including those based on technological developments [1], notably in the field of engineering, where applications are being developed that guide human-AI collaboration [2]. The data sources for these models are varied and include those based on text and images [3] and natural language processing (NLP) [4].

The advantages of its use range from the simplification of information for non-expert population groups [5], interpretability of complex multimodal data [6], support for research in relation to the correlation of keywords and citations [7] and robotic manipulation, where LLMs improve the reasoning and the robot's ability to handle increasingly complex tasks [8].

In this field of robotics and electronic systems, LLMs allow applications such as the assignment of tasks to unmanned aerial vehicles using textual format [9], which is also used in terrestrial mobile robotic planning systems [10] and manipulation of fixed robots such as anthropomorphic arms using visual feedback [11] that improve human-robot interaction [12].

However, in human-robot interaction, effective communication is necessary to guarantee precision and coherence in the robotic task [13], which has been addressed from the robot's interaction with its environment as referred to in the previously mentioned state of the art. As a complement and contribution to this work, the development of a framework based on natural language processing for the action of electronic control systems for fixed or mobile robots is presented, where the use of a long language model in the textual understanding of user settings is evaluated, in this case the configuration thresholds of distance alert systems based on ultrasonic sensors, as a voice-language-action model in human-robot interaction systems.

Section II outlines the methodology employed; Section III presents the results obtained and their analysis; and finally, Section IV sets out the study's conclusions

II. Methodology

The objective is to achieve human-machine interaction via user voice commands, using NLP to adjust the proximity alert generation threshold based on distance readings from an ultrasonic sensor. The audio input is processed locally on a PC using Whisper, and the resulting output is sent to the LLM Qwen 3.5, which is integrated using Python to interpret the distance alert threshold adjustment command. This command is then sent via serial communication to a STM32 microcontroller, which reads an ultrasonic sensor and generates the corresponding alert, as illustrated in Fig. 1.

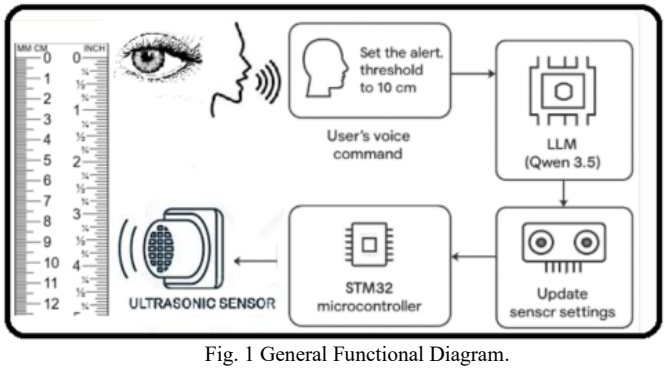


Fig. 1 General Functional Diagram.

The methodology corresponds to applied research aimed at implementing robotic control commands in a natural way for the user; Fig. 2 illustrates the steps used described below.

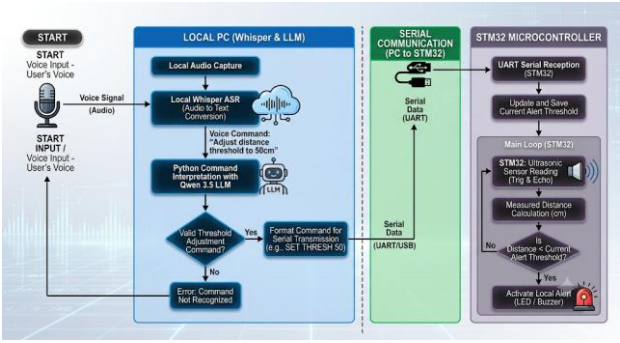


Fig. 2 Descriptive diagram of how the system works.

Fig.3 shows the setup of the system developed for this research. Additionally, Fig. 4 shows the computing hardware running the LLM and Python code, along with the serial connection to the STM32 microcontroller, is visible. An ultrasonic sensor is connected to the microcontroller, and visual feedback is provided via colored LEDs that indicate distance segments in 10 cm increments.

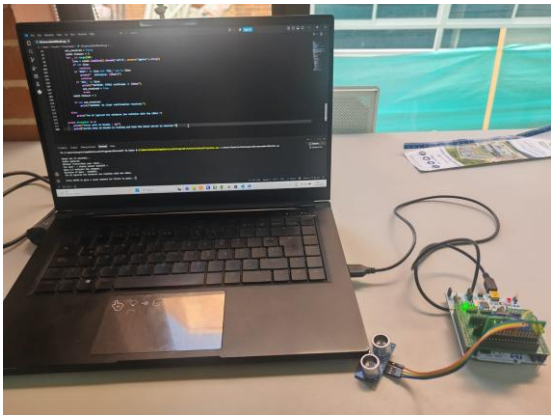


Fig. 3 Setup of the experimental system.

The HC-SR04 ultrasonic sensor is illustrated in Fig. 4, this is a low-cost ranging sensor widely employed in embedded systems and robotic applications. Table I presents the characteristics of the sensor. The HC-SR04 determines the distance to an object using the time-of-flight principle, it is the measurement of signal departure and return times to and from the microcontroller, by transmitting a 40 kHz ultrasonic pulse. The sensor provides a measurement range from 2 cm to 400 cm with a typical accuracy of ±3 mm under favorable environmental conditions.



Fig. 4 Sensor HC-SR04.

TABLE I
Sensor HC-SR04 Characteristics

Parameter	Specification
Sensor Model	HC-SR04 Ultrasonic Distance Sensor
Measurement Principle	Ultrasonic pulse echo (Time-of-Flight)
Operating Voltage	5 V DC
Operating Current	Approximately 15 mA
Standby Current	Less than 2 mA
Operating Frequency	40 kHz
Measurement Range	2 cm to 400 cm
Measurement Resolution	Approximately 3 mm
Measurement Accuracy	±3 mm (under ideal conditions)
Measuring Angle	Less than 15°
Trigger Signal	10 µs TTL pulse
Echo Output	TTL pulse proportional to measured distance
Response Time	Approximately 38 ms maximum measurement cycle
Operating Temperature	−15 °C to +70 °C
Dimensions	45 × 20 × 15 mm
Weight	Approximately 9 g
Interface	4-pin (VCC, Trig, Echo, GND)
Typical Applications	Distance measurement, obstacle detection, mobile robots, automation systems, autonomous vehicles, and IoT projects

The input system is based on voice commands where the user indicates which alert threshold to configure; this voice command then enters a speech to text (STT) algorithm, for the case based on Whisper [14]. A fundamental basis of the development is the user's ability to interact naturally; Whisper was selected for this task because it enables audio preprocessing based on normalized, fixed blocks divided into continuous 30-second segments. From these, it constructs a Log-Mel spectrogram as a visual representation of frequencies over time that simulates how the human ear perceives sound pitch and volume. The algorithm relies on token-per-word prediction and employs a cross-attention layer to ensure the predicted text aligns with the received audio frequencies.

This STT employs Fourier transform as is shown in (1), which converts the audio signal in the time domain to the frequency domain for digital processing in spectrograms that visually illustrate how these voice command frequencies change over time

$$X(f) = \int_{-\infty}^{\infty} x(t)\, e^{-j\omega t} dt \qquad (1)$$

The spectrograms of the command are used to tokenize the text, where each word is converted into a token using in (2), yielding a number representative of that unit of text, where the function used (f) is a mapping of words to tokens.

$$Token = f(words) \qquad (2)$$

Whisper decodes the audio representation into text by calculating the probability that the sequence of tokens corresponds to the target audio input, using the beam search algorithm by (3), where y is the sequence of tokens (text) and x is the user's audio representation.

$$\hat{y} = arg\ max_{v} P(y|x) \qquad (3)$$

To ensure the system runs smoothly, the architecture has been divided into two processing units that work in parallel, Fig. 5 shows that general architecture.

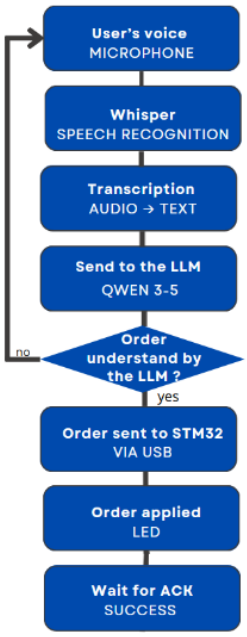


Fig. 5 General arquitecture of the system.

Firstly, the computer runs the Python script that handling AI. As shown in Algorithm 1, this script records the user's voice with the microphone, then uses Whisper to transcribe that speech into text. Once the transcription is complete, the system is called Qwen 3-5. Qwen 3.5 was selected as a family of open-source language models featuring native multimodal architecture; it is optimized to run locally on consumer hardware without cloud dependency and requires no external adapters to read screenshots, interpret diagrams, or perform optical character recognition (OCR). It utilizes a native 256K context window, enabling the analysis of extensive documents or entire codebases directly in local memory. Of the two configurable reasoning modes supporting deep reasoning and Fast (for immediate, low-latency responses), the model was used in fast mode, as the reasoning process is unnecessary and consumes more time. Multilingual support is crucial for the application; Qwen 3.5 supports over 200 languages and dialects while maintaining high accuracy in Spanish, the authors' native language.

Using a prompt, the LLM analyzes the sentence to determine the action to be performed and the parameters to change (LED color and distance). If the voice command is inaudible or misunderstood, the script ignores the instruction and waits for a new voice request.

However, if the intent is clear, the script translates it into a command for the microcontroller (for example: LIMIT_B:150). Once this command is extracted, it's sent to the microcontroller via the USB port. The computer then waits for an acknowledgment (ACK) confirming the success of the operation.

```
while hardware system is active do
    Capture user's voice input via microphone
    Input: Incoming USB serial data
    Read incoming byte from serial port
    Transcribe audio to text using Whisper
    Communicate with Qwen 3.5 LLM to extract parameter
    Receive answer from LLM and transmit via USB
    Initialize command buffer on STM32 microcontroller
    if byte indicates end-of-line of the LLM command
     then
        Execute the received answer to validate new thresholds
        Update LED actuators state based on new limits
        Send acknowledgment (ACK) back to computer
        Clear command buffer
    else
        Stop process and go back to the initial state of code
    end
end
```

Algorithm 1: Data capture, translation using an LLM, and sending commands to the STM32.

Secondly, as shown in Algorithm 2, the STM32 process begins with device initialization and interrupt activation (Timer), ensuring asynchronous reception. The system then enters an execution loop where it interrogates the ultrasonic sensor via a 10µs pulse and calculates the distance in millimeters. Once the distance is achieved, the logic resets the state of the LEDs before setting up conditions. The light modules are activated gradually: green, red, blue and yellow as the obstacle crosses the defined limits. Finally, the microcontroller transmits the distance it has just measured to the computer. If, at the same time, the system detects that it has finished receiving a new voice command, it will immediately process it. It then updates its new safety thresholds in memory, sends a confirmation message (ACK) to the computer to verify that the command has been successfully executed, and then repeats this observation cycle.

```
Initialize hardware peripherals
Transmit initialization status and enable UART receive interrupt
while true do
  Capture user's voice input via microphone
  Input: Ultrasonic sensor signal
  Trigger ultrasonic sensor with 10 µs pulse
  Measure Echo pulse duration using hardware Timer
  if timeout occurred then
     distance_mm = saturate limit;
     else
     distance_mm = echo_duration * 0.34) / 2.0;
```

```
            Turn OFF all LEDs
        end
        if human_detected then
        Turn ON all LEDs
        else if distance_mm &gt; 0 and distance_mm &lt;= limit_green then
        Turn ON Green LEDs
        else if distance_mm &gt; limit_green and distance_mm &lt;=
                limit_red then
                Turn ON Green and Red LEDs
        else if distance_mm &gt; limit_red and distance_mm &lt;=
        limit_blue then
        Turn ON Green, Red, and Blue LEDs
        else
        Turn ON all LEDs
      end
        Transmit calculated distance_mm to PC via USB
    if command_ready then
        Parse UART buffer for LIMIT_G, LIMIT_R, LIMIT_B, or
        YOLO_H
        Update respective distance limit
        Transmit acknowledgment ACK to PC via UART
    end
end
```

Algorithm 2: Control of the alarm system via commands received from the computer

To measure the distance between the robot and an obstacle, the STM32 microcontroller controls the ultrasonic sensor. The main program generates an electrical signal (Trigger) that triggers the emission of sound. The system then listens to the returning wave (Echo) and measures the elapsed time using a timer.

To convert this time into a distance in millimeters, the speed of sound in air (0.34 mm/µs) is multiplied by the time, then divided by two to account for the two-way travel of the sound wave, as is describes in (4).

$$(d = \frac{t \times 0.34}{2}) \quad (4)$$

III. Analysis and Results

To verify that the project works under real-world conditions, testing sessions were conducted with four different users in three languages: English, French, and Spanish. While they were giving commands, the Python script recorded all data in the background, such as the number of tokens used per voice command, the elapsed time, the number of voice commands issued by each user, and the language used to give the command.

The first success of these tests is the system's flexibility. Whether the user spoke in English, Spanish, or French, and the system was able to extract the correct intent and adjust the thresholds on the STM32 board.

However, it became evident that the AI's performance (response time and number of tokens consumed) varied from one person to another. For example, as shown in Table II, user Adam consumed significantly more resources and time (an average of 1409 tokens with a response time of 14.4 seconds per command) than Ricardo, who was much more efficient (an average of 761 tokens with a response time of 6.8 seconds per command).

TABLE II
TEST METRICS RELATED TO MULTIPLE USERS

User	Total Cmds	Avg Time (s)	Avg Tokens	Languages Used
Adam	10	14.4	1409.7	DE, FR, KO
Fabio	8	11.8	1194.1	EN, BG, ES, TR
Jose	15	14.2	1053.1	EN, ES, AR
Ricard	8	6.8	761.0	EL, ID, ES, AR

These higher power consumption values were due to different factors: First, when the Whisper model has difficulty understanding a voice command, there may be a language misinterpretation, which results in the detection of a language that was not actually spoken; consequently, the command is incorrect and is not understood by the LLM. Voice detection then failed, recording false values that were not actually spoken. This is also why other languages were detected and reported in the logs even though we had only spoken French, English, and Spanish.

Robustness against voice variations is provided by the chosen LLM, based on the attention mechanism of the Transformer architecture that comprises it [15]. Table III illustrates the characteristics of the model used.

TABLE III
TEST METRICS RELATED TO MULTIPLE USERS

Feature	**Description**
Model Name	Qwen 3.5
Developer	Alibaba Cloud – Qwen Team
Model Type	Large Language Model (LLM) with native multimodal capabilities
Architecture	Hybrid Attention + Sparse Mixture-of-Experts (MoE)
Largest Open Model	Qwen3.5-397B-A17B (397B total parameters, 17B active parameters)
Context Window	Up to 262K tokens (extendable to 1M tokens in hosted versions)
Supported Modalities	Text, images, documents, charts, audio, and video (Omni models)
Main Capabilities	Natural language understanding, reasoning, code generation, mathematical problem solving, document analysis, agent-based task execution, multimodal understanding, and function calling
Programming Languages	Python, C/C++, Java, JavaScript, TypeScript, Go, Rust, SQL, HTML/CSS, Bash, MATLAB, R, PHP, Swift, Kotlin, among others
License	Open-weight models available under Apache 2.0 (selected versions)
Typical Applications	Conversational AI, intelligent agents, software development, data analysis, robotics, industrial automation, education, and scientific research.

The measured times are based on the use of a computer system equipped with an Intel Core i7 processor and an 8 GB NVIDIA RTX 3070 GPU.

Fig. 6 shows some performance tests conducted using the system, A set of different-colored LEDs is used to identify measurement ranges and visually confirm the settings based on the user's voice commands. For example, the user requests to adjust the distance threshold for each LED panel color—blue, green, red, and yellow.

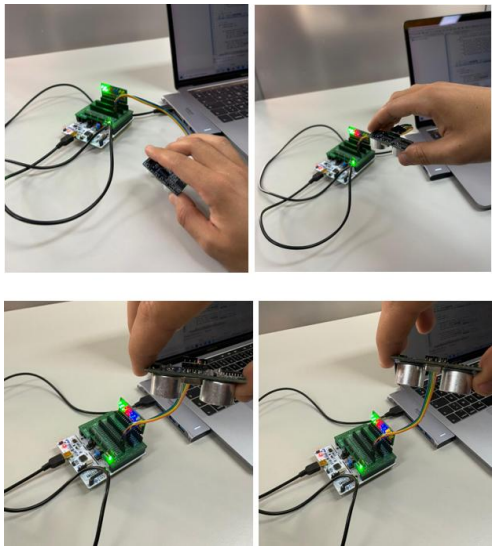


Fig. 6 Sensor HC-SR04.

These differences in resource consumption may also be due to variations in the complexity of the sentences spoken by different users. In fact, when users use relatively long sentences such as, “Set the blue light limit to 15 centimeters,” or relatively short sentences such as, “Blue at 150,” as shown in Fig. 7 we observe a significant difference in resource consumption, despite a similar result. In fact, the shorter and more concise the sentence, the lower the resource consumption, which explains part of the difference in resource consumption among users.

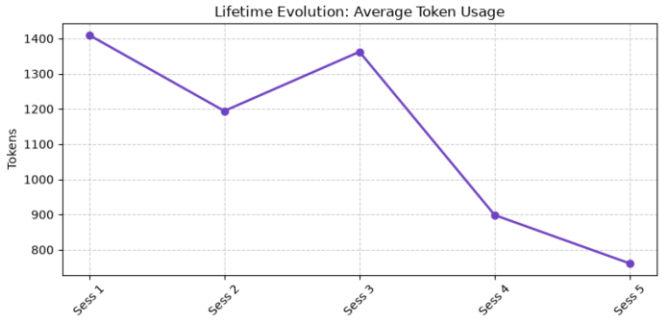


Fig. 7 Trend in Average Token Usage per Session Over Time.

IV. Conclusions

The proposed architecture allows for the decoupling of computationally intensive AI processing from embedded control, with audio capture, local transcription using Whisper, and intent extraction using the Qwen 3.5 model all performed on a computer, while the STM32 microcontroller is responsible solely for measuring the distance from the ultrasonic sensor and generating alert signals on an LED board, resulting in a more natural, intuitive, and flexible robotic control interface.

The designed voice interaction system achieved a total of 41 historical commands, with an overall average time of 12.3 s and an average consumption of 1110.6 tokens per interaction, figures that quantify the typical computational load of the audio–Whisper–LLM–STM32 pipeline. Furthermore, the analysis by operator reveals clear differences in performance: while Adam and Jose have average times exceeding 140 s and consumption of more than 1,050 tokens per command, Ricardo’s figures are 6.8 s and 761.0 tokens, indicating that both the speaker’s characteristics and the command formulation significantly influence the efficiency of recognition and semantic interpretation.

Despite the variability among multiple users, the results show that the system maintains the ability to operate with multiple languages and multiple operators, with response times in the order of tens of seconds and a consumption of between 760 and 1,400 tokens per command, parameters that constitute a useful quantitative benchmark for future comparisons with other voice control schemes in robotics.

The results support the feasibility of using large, locally executed language models as a bridge between a user’s voice and the control of physical parameters in robotics, thereby opening a line of research toward more human-like multimodal interfaces for autonomous systems and embedded platforms.

Acknowledgment

Product derived from the research project IMP-ING-4307 entitled “Design of a mobile intelligent agent for residential use for basic care of people, based on a visual description model” funded by the vice-rectorate of research of the Universidad Militar Nueva Granada, year 2026.

References

[1] D. Zhai, K. Zhao, M. Wang, L. Zhai, & S. Xu. “AI-driven opportunity forecasting for technology startup identification: Integrating graph embedding, LLMs, and informetric analysis”. *Technological Forecasting and Social Change*, 227, 124649. 2026.

[2] Z. Xia, B. Zhong, S. Zhang, T. Zhao & M. J. “Skibniewski. Graph-driven embedding reinforcement and traceable LLM agent for reliable element alignment in construction report generation”. *Automation in Construction*, 183, 106816, 2026.

[3] P. G. A. Njock, Z.-Y. Yin, H.-R. Xu, & N. Zhang. “A multimodal retrieval-augmented framework for geotechnical soil classification using cross vision-text embeddings and LLM reasoning”. *Advanced Engineering Informatics*, 76(Part A), 104971. 2026.

[4] A. Ghourabi, H. Chouaib. “A Ransomware Detection Approach Based on LLM Embedding and Ensemble Learning”. *Computers, Materials & Continua*, 87(1), 98. 2026.

[5] P. Esmaeilzadeh. “Embedding LLMs in the patient portal to summarize acute minor illness information: A three-arm experimental study”. *International Journal of Medical Informatics*, 219, 106547. 2026.
[6] G. Petnehazi, & B. Aradi. “HERCULES: Hierarchical embedding-based recursive clustering using LLMs for efficient summarization”. *International Journal of Cognitive Computing in Engineering*, 7, 445–456. 2026.
[7] X. Hu, H. Gu, & Y. Tang. “An embedding-based approach for measuring science–technology topic linkages using LLMs”. *Information Processing & Management*, 64(1, Part A), 104983. 2027.
[8] K. Peng, Q. Li, Z. He, B. Zhang, X. Fu, B. Li, X. Wang, Z.-Q. Cheng, Y. Yan, & X. Peng. “A survey of robotic manipulation: From bottom-up approaches to end-to-end paradigms with LLMs”. *Neurocomputing*, 674, 132921. 2026.
[9] B. Wu, & R. Xiao. “Evolutionary attraction–repulsion algorithm embedded with LLM for UAV task allocation”. *Advanced Engineering Informatics*, 66, 103428. 2025.
[10] F. Favali, L. Sabattini, & V. Villani. “LLM-based reasoning for robotic planning: Robustness to task and environmental complexity*”. Robotics and Autonomous Systems*, 204, 105565. 2026
[11] G. Yi, H. Chu, S. Wen, & F. Wang. “Enhancing stability and reliability in LLM-driven robotic manipulation through human skill demonstration and visual tracking”. *Biomimetic Intelligence and Robotics*, 100305. 2026.
[12] S. Liu, M. Liu, Z. Ming, Y. Huang, L. Ma, D. Zhang, Y. Song, J. Zhang, & T. Yan. “Copilot: A framework for integrating LLM and BMI to enhance human–robot interaction”. *Robotics and Computer-Integrated Manufacturing*, 101, 103291. 2026.
[13] L. Fernández-Becerra, A. M. Guerrero-Higueras, F. J. Rodríguez-Lera, & V. M. Olivera. “Generating trustworthy and context-aware explanations for autonomous robots using an LLM agent-based RAG architecture”. *Expert Systems with Applications*, 325, 132631. 2026.
[14] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, & I. Sutskever. “Robust speech recognition via large-scale weak supervision”. *In Proceedings of the 40th International Conference on Machine Learning,* pp. 28492–28518 2026.
[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser & I. Polosukhin. “Attention is all you need”. *In Advances in Neural Information Processing Systems*. pp. 5998–6008. 2017.