Voice Recognition Alphabot

Project developed in the context of the Curricular Unit SETR 2022/2023

André Pereira
up201905650

Beatriz Aguiar
up201906230

João Marinho
up201905952

Tiago Silva
up201906045

***Abstract*—This report presents the implementation of a voice-controlled Alphabot for the Embedded and Real-Time Systems curricular unit project. The system utilizes the Alphabot II platform and Raspberry Pi 4 Model B for hardware integration. To address the real time constrains of our system and incorporate the microphone, a preemptive task scheduler was used. This allows the robot to accurately interprets voice commands and take movement control actions accordingly. A collision detection mechanism ensures safety by halting the robot's movement upon detecting nearby objects. The report outlines the project's objectives, software architecture, results, and assessment of complexity.**

I. Introduction

A. Background and Motivation

The project focuses on implementing a real time embedded system featuring a voice-controlled Alphabot. The objective is to explore the capabilities of voice-controlled robotics and develop a system that accurately interprets voice commands for controlling the robot's movement. The motivation behind this project is to enhance human-robot interaction and make robotics more accessible through a natural and intuitive interface.

B. Objectives of the Project

In the context of the curricular unit project, our primary objective is to implement a real time voice-controlled Alphabot, driven by the mentioned demands. The specific objectives of the project are as follows:

Voice recognition system
Implement an efficient voice recognition system capable of accurately interpreting voice commands to control the Alphabot's movement in multiple directions. This involves exploring and integrating suitable voice recognition algorithms and techniques.

Preemptive task scheduling
Design and implement a preemptive task scheduling mechanism to efficiently manage the concurrent execution of motor control, voice recognition, and sensor tasks. This ensures timely response to voice commands and seamless multitasking capability.

Collision detection mechanism
Integrate a collision detection mechanism to enhance the safety of the Alphabot. This involves incorporating sensors to detect nearby objects and automatically halting the robot's movement to prevent collisions.

II. System Design

A. Hardware

1) Alphabot: The Alphabot II serves as the primary physical structure and mobility platform for our voice-controlled system. The robot incorporates diverse actuators and sensors to enable versatile functionality. The key components used in our project are the following:

Motor Control
The TB6612FNG dual H-bridge motor driver is utilized for controlling the Alphabot's motors. This motor driver provides the necessary power and control signals to drive the wheels of the robot.

IR Sensors
Two ST188 reflective infrared photoelectric sensors are employed for obstacle avoidance. These sensors detect objects in the robot's vicinity and provide essential input for collision prevention.
Two potentiometers are integrated into the system to adjust the obstacle avoidance range. These allow for fine-tuning the distance at which the robot perceives obstacles.

Indicators
Two obstacle avoiding indicators are incorporated into the system to provide visual feedback on the robot's proximity to obstacles. These indicators utilize integrated LEDs that illuminate based on the robot's distance to obstacles, thereby enhancing user awareness of potential collisions.

Power Supply
A battery holder compatible with 14500 batteries is employed to provide power to the voice-controlled Alphabot system. These batteries offer a reliable and portable power source for the robot.

2) Raspberry Pi: The Raspberry Pi 4 Model B was chosen as the central component for the AlphaBot system due to its powerful architecture and versatility. With its capable processor and sufficient memory, the Raspberry Pi 4B provided the necessary computational resources to handle the execution of multiple tasks.

The integration of the Raspberry Pi into the AlphaBot involved connecting it to the robot's hardware components and sensors.

3) Microphone: To enable audio input and extract commands from the environment, we integrated a USB microphone with the Raspberry Pi by utilizing one of its available USB ports. This straightforward connection provided us with the capability to capture and process audio signals seamlessly.

B. Software

1) Preemptive Scheduler: The system incorporates a preemptive scheduling mechanism with fixed priorities, where the priorities are determined according to the importance of each task.

2) Tasks: The scheduler module in the AlphaBot system coordinates the execution of several distinct tasks, each responsible for specific functionality.

Motor Task
The Motor task controls the movements of the AlphaBot, enabling it to move in different directions. It receives commands from higher-level modules and translates them into precise motor control actions.

Voice Task
The Voice task is responsible for processing audio input from the USB microphone and extracting voice commands from the environment. It translates the recognized commands into corresponding actions that can be executed by the AlphaBot.

Sensor Task
The Sensor task is responsible for detecting obstacles and ensuring the safety of the AlphaBot. It continuously monitors the surroundings of the robot and triggers the appropriate action, stopping, when obstacles are detected within the specified range.

III. Implementation

A. Integration of Hardware and Software Components

The integration of hardware and software components in the AlphaBot system posed several challenges during the development process, ultimately leading to delays in the overall progress of the project.

Initially, we were provided with a Raspberry Pi 3 Model B as the main computing platform. However, we encountered a hurdle when attempting to control the movement of the AlphaBot's wheels. The PIN GPIO numbers provided in the tutorial [1] were specific to the Raspberry Pi 4, and it took us a substantial amount of time to find the appropriate pinout configuration that aligned with our Raspberry Pi. This unexpected setback consumed valuable development time and required extensive troubleshooting and research.

Once we finally resolved the pinout issue and successfully established the connection for controlling the wheels, we encountered another setback. It became apparent that the provided AlphaBot sensors were damaged and not functioning correctly. This unexpected discovery necessitated a change in our approach, and we had to switch between multiple AlphaBot components to find functional replacements.

Furthermore, we encountered a recurring obstacle in understanding the pin configurations required for controlling the Alphabot. Despite our best efforts, we were unable to determine the correct pin assignments.

Fortunately, a breakthrough occurred when we were provided with a Raspberry Pi 4 to test. The default PIN numbers outlined in the manual were fully functional, enabling us to establish the necessary connections and successfully control the Alphabot's movements. Additionally, the replacement sensors operated as expected, providing accurate feedback to the system.

B. Scheduler

The scheduler was implemented in C++, a language known for its efficiency and high performance. We took advantage of the code provided by the curricular unit professors as a starting point for our implementation.

The scheduler consists of two main functions: *scheduler_schedule* and *scheduler_dispatch*. These functions are called every tick, 0.3 seconds, working together to ensure the execution of tasks in a timely manner.

The *scheduler_schedule* function is responsible for determining which tasks are eligible for execution at a given time, setting the appropriate tasks as ready to execute.

On the other hand, the *scheduler_dispatch* function is responsible for invoking the task function when a task is ready to be executed. In our preemptive system, interrupts are enabled once a task starts running. This allows higher-priority tasks to interrupt the current task and take control, ensuring that critical tasks are executed promptly.

C. Tasks

The tasks, Table I, are represented as C++ structures, each with a period, delay, and execution function. These structures encapsulate the properties and behaviors of each task. Additionally, each task is associated with a Python module, responsible for task-specific logic and functionality. The decision to use Python modules was driven by factors such as the extensive libraries and frameworks available, as well as the strong support from AlphaBot's manual [1], which facilitated the development of task-specific logic.

Furthermore, each module utilizes a command buffer for seamless communication and coordination between tasks. This modular design approach enables efficient task execution and system integration.

Task	Priority	Period (interrupts)	Delay (interrupts)
Motor Task	1	1	1
Sensor Task	2	1	2
Voice Task	3	14	3

TABLE I
Tasks' Details

D. Motor Module

The motor module plays a crucial role in the AlphaBot system by invoking the relevant motion control functions. The motor task is responsible for processing the command buffers and determining the appropriate command to execute based on buffer priority. The task ensures that the sensor buffer and voice buffer are emptied and assigns the corresponding commands for execution, giving higher priority to the commands from the sensor buffer in order to avoid collisions.

E. Voice Module

To enable voice control, the AlphaBot integrates a speech recognition module developed using a Python library [2], which is an integral part of the overall system.

This module contains a function called *voiceCommand* that utilizes the microphone to listen for a duration of 1.5 seconds. Prior to capturing audio, the microphone is adjusted to account for ambient noise, ensuring reliable and noise-free audio capture. Subsequently, the resulting audio value is sent to the *Google Speech Recognition API* [3] to obtain potential transcriptions of the audio. These transcriptions are then processed using regular expressions, regex, to determine if they match any of the predefined patterns: $"(turn)?left"$, $"(turn)?right"$, $"(move)?forward"$, $"(move)?back"$, $"stop"$, $"talk"$ and $"kill"$. If a match is found, the task sets its buffer according to the respective action enabling the robot to execute the appropriate movement or behavior.

The robot's available actions are as follows:

- **forward** Move forward.
- **backward** Move backward.
- **left** Turn left.
- **right** Turn right.
- **stop** Stop the movement.
- **talk** Emit a buzz.
- **kill** Terminate the process.

1) Sensor Module: The sensor module is responsible for verifying inputs from sensors, detecting objects, and assigning the stop command to its buffer in case of object detection.

IV. System Results

A. Schedulability Analysis

We employed a comprehensive measurement approach to evaluate the performance of the system. We measured the worst-case execution time and worst-case response times of the tasks through a series of 50 measurements, Table II, ensuring statistical significance and reliability of the data.

Task	WCET(µs)	WCRT(µs)
Motor Task	15	115
Sensor Task	409	1164
Voice Task	3232015	4126404

TABLE II
TASKS' MEASURED TIMES

Additionally, we calculated the theoretical values of the worst-case response times based on the system's parameters and characteristics. These theoretical values served as a benchmark for comparison with the measured worst-case response times.

- T1 - Motor Task:

$$Rwc_1(0) = 15\mu s$$

- T2 - Sensor Task:

$$Rwc_2(0) = 409 + 15 = 424\mu s$$
$$Rwc_2(1) = 409 + 15 * \lceil\frac{424}{300000}\rceil = 424\mu s$$

- T3 - Voice Task:

$$Rwc_3(0) = 3232015 + 409 + 15 = 3232439\mu s$$
$$Rwc_3(1) = 3232015 + 409 * \lceil\frac{3232439}{300000}\rceil + 15 * \lceil\frac{3232439}{300000}\rceil = 3236679\mu s$$
$$Rwc_3(2) = 3232015 + 409 * \lceil\frac{3236679}{300000}\rceil + 15 * \lceil\frac{3236679}{300000}\rceil = 3236679\mu s$$

The attained practical values surpass the theoretical worst-case scenarios, therefore we will consider the first ones in our *Response time-based schedulability test* against the task deadlines, which, in this case, correspond to the respective task period.

We can confidently conclude that our system is schedulable, given that the practical worst-case response times are smaller than the deadlines.

B. Practical Outcome

By applying the knowledge acquired in our class, we have successfully implemented voice control in the AlphaBot project. With our understanding of real-time systems and scheduling algorithms, we ensured prompt command detection and accurate translation into robot actions.

V. Conclusion

The main objective of this project was to develop a real-time voice-controlled Alphabot.

In order to accomplish this objective, we successfully implemented a preemptive scheduler that handles three specific tasks: motor, sensor, and voice. Through the seamless execution of these tasks, users are able to command the robot to perform desired actions.

For future enhancements to the project, we propose replacing the current USB microphone with a wireless alternative. This modification would enable users to remain stationary while issuing commands to the robot, enhancing convenience and ease of use.

VI. Self-assessment

Each member contributed equally to the project outcome.

References

[1] Waveshare Wiki, AlphaBot2-PI, [Online]. Available: https://www.waveshare.com/wiki/AlphaBot2-Pi. [Accessed: 6-May-2023].
[2] SpeechRecognition 3.10.0, [Online]. Available: https://pypi.org/project/SpeechRecognition/. [Accessed: 6-May-2023].
[3] Speech-to-Text V2, [Online]. Available: https://cloud.google.com/speech-to-text. [Accessed:]