Warthog Robotics @Home 2025 Team Description Paper

Rhayna Casado[1,2], Kenzo Sakiyama[1], Eduardo Rodrigues[1], João Gomes[1,2],
Rafael Lang[2], Ivan da Silva[2], Roseli Romero[1]
[1]*Institute of Mathematics and Computer Science, University of São Paulo*
[2]*São Carlos College of Engineering, University of São Paulo*
São Carlos, São Paulo, Brazil
Team Website: www.wr.sc.usp.br
Corresponding Author: rhayna.casado@wr.sc.usp.br

***Abstract*—This paper describes the robot Antares, Warthog Robotics @Home project, focusing on innovations for RoboCup @Home Brazil Open 2025. The robot and its auxiliary software and tools have been designed inside the Robotics Center (CRob) of the University of São Paulo (USP) for the past seven years as scientific development to help disabled people in household environments. The current development was centered on system rebuilding, as the old ones were legacy and outdated. Therefore, we focused on conceiving a parallel architecture and open-source update of ROS2 packages, along with model selection for human-robot and robot-environment interaction and a new robotic manipulator. It states that the improvements made to the robot enable it to test new technologies and conduct more research on them, thereby evolving to develop the league's tasks and contributing to domestic and assistive robot research. Also, this paper reinforces future projects to improve Antares' performance in accomplishing household tasks.**

***Index Terms*—RoboCup @Home Brazil Open, ROS2, Software Architecture Overhaul.**

I. Overview

Warthog Robotics is a traditional research and extension robotics group from the University of São Paulo (USP). Since its creation in 2007, the group has aimed to research and develop robotics-related technologies and their application in complex competitive environments. Throughout the years, the group has participated in more than eight robotics leagues, achieved over 30 prizes, and published more than 60 papers.

In 2017, Warthog Robotics launched its assistance and service robot program with robot Antares, designed to explore and advance human-robot interaction and develop practical robot household skills. Desiring to contribute to the scientific and domestic robotics community, the Warthog Robotics @Home team has been effortlessly researching areas like machine learning, path planning, and control.

Since the beginning of the robot development, Antares has been participating in the RoboCup@Home league in the Latin American Robotics Competition (LARC) and RoboCup Brazil Open Competition (CBR). We have achieved 1st and 2nd places in the previous years and contributed to the scientific community with studies in navigation systems, control, emotions simulations, docker architecture, object detection, face detection with mask, gesture recognition, and semantic mapping.

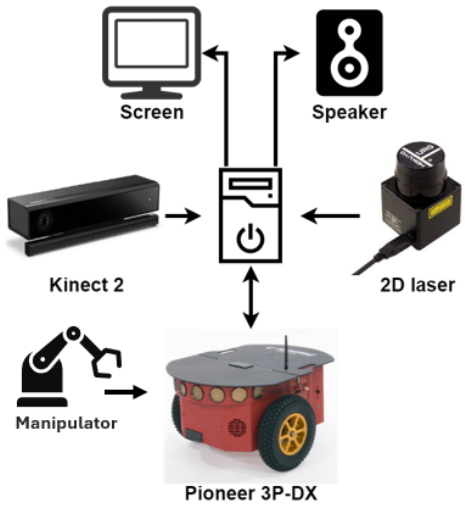

Fig. 1. Overview of the robot Antares' sensors and actuators.

However, in a post-pandemic scenario, the team was renewed, as all the past members quit the group, leaving a legacy program with little documentation and an outdated approaches. Thus, we decided to rebuild our system to give a fresh and updated start to robot Antares, enabling our development to evolve from local to international competitions.

II. Antares

Robot Antares was developed as an assistive robot for people in their household environment, seeking scientific contribution to human-robot and robot-environment interaction and social skills for robots. It has a Pioneer 3-DX as its hardware platform, serving for locomotion and power supply. It has a LiDAR and a camera sensor, an LCD screen, external speakers, a microphone, and a computer to process and integrate the equipment's functions and environment interactions. The mechanical structure is aluminum and plastic-based and designed and manufactured for Antares. An overview of the Antares modules is presented in Figure 1.

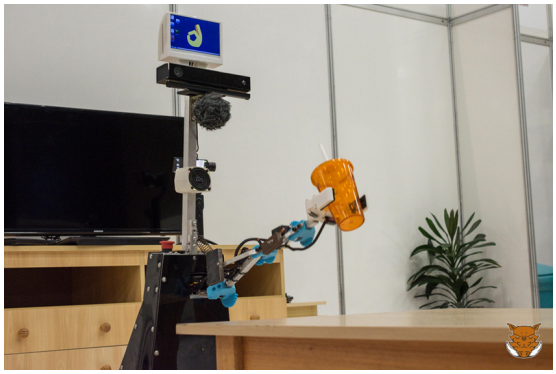

Fig. 2. Robot Antares' previous manipulator.

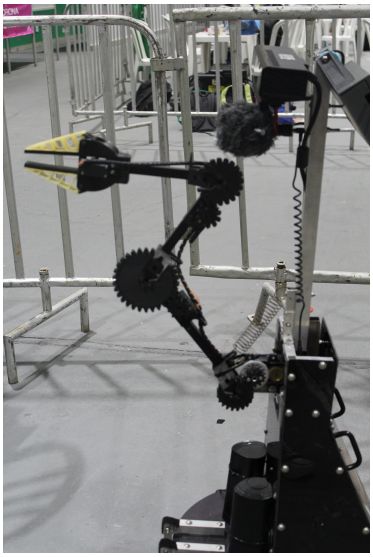

Fig. 3. Robot Antares' newer manipulator.

A. Manipulator

The development of a robotic manipulator has been the main project of the hardware team throughout the past year. To have a better gripper and payload than the previous version, presented in Figure 2, we focused on developing a better gripper and increasing torque and range, avoiding weight gain so there is no prejudice on the robot's center of gravity.

Also, for better resource management and performance, we designed a printed circuit board (PCB) for motor control, sensor feedback, and software communication for the manipulator. The PCB drives the power supply for the motors and gripper's distance sensor and interfaces the communication with the mini PC through a Raspberry Pi Pico board.

As an outcome, shown in Figure 3, we developed a 3 degrees of freedom (DOF) manipulator using aluminum and 3D-printed polylactic acid (PLA) for the main structure, a pliable 3D-printed gripper, using only servo motors and equipped with a camera and a distance sensor. With a total weight of 2kg and a horizontal range of 70cm, the robot can pick objects up to 500g from the floor to 80cm above the ground.

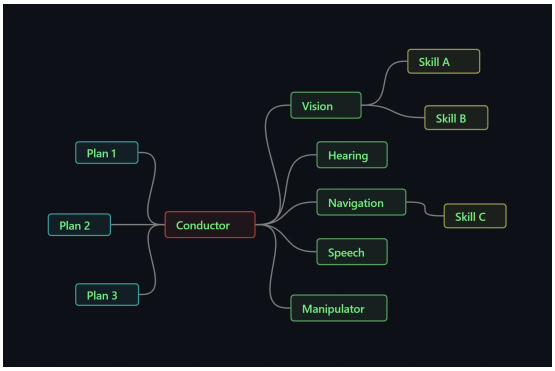

Fig. 4. Overview of the components of the new software architecture.

Furthermore, for more management and determinism in the manipulator control, we opted to use FreeRTOS [1] in our firmware, a real-time operating system (RTOS) designed to multitask, be small, have fast execution time, and run on micro-controllers. For developing the manipulator control, we integrated it with our software architecture on Robot Operating System 2 (ROS2) [2] using the MoveIt 2 [3] platform.

III. Software Architecture Overhaul

Due to a legacy system with a lot of depreciated or outdated packages, Antares underwent a significant overhaul in its software architecture, moving away from a state machine-based structure to a more robust and flexible parallel architecture allows for more efficient and responsive behavior from the robot, enabling it to execute multiple tasks simultaneously without the bottleneck limitations inherent in the previous state-based system. The parallelism now effectively manages the robot's subsystems, including navigation, manipulation, and human-robot interaction, allowing for better scalability and maintainability of the software. An overview of the new software architecture is presented in the Figure 4.

The old software infrastructure was a collection of ROS nodes that followed the publisher/subscriber protocol and enabled easy integration and communication between different modules. With the overhaul in mind, the old nodes were updated to ROS2. In addition, to improve software modularity and compatibility (for future updates), the nodes are being implemented as Docker [4] containers, which allows for easier local development and enables the group to quickly remove a node from the robot and run it on separate hardware to save energy and computational resources.

The components present in Figure 4 will be described as follows:

- **Skills:** correspond to low-level actions, such as 'go to ...', 'listen to ...', 'find the object ...', etc.
- **Managers (Vision, Navigation, Speak, Listener, and Arm):** managers customize the skills behaviors with custom arguments and queue the requests from the conductors.
- **Conductor:** is the component that communicates with the managers and the plans, forwarding messages between

the two components. Conductors redirect the requests from the plans to the suitable managers (vision, navigation, etc).
- **Plans:** implement high-level planning for complex tasks. The plans coordinate skills to perform the competition tasks.

The information and requests flow from the plans to skills (and vice-versa) like a client-server model. Plans act as clients and the managers and conductors act as servers.

A. Docker

To facilitate deployment and manage dependencies effectively, we leveraged Docker to containerize different system parts. This containerization significantly simplified the setup process, reduced compatibility issues, and improved reproducibility, making the system easier to develop, test, and deploy in competition environments. Docker also allowed for easier collaboration among team members, ensuring that each component could be tested and developed independently within its environment.

Additionally, we utilized Docker Compose to orchestrate the multiple containers, streamlining the management of interconnected services. This tool enabled us to define and run multi-container applications using a single YAML configuration file, simplifying the process of configuring, starting, and stopping the system as a cohesive unit. Docker Compose ensured that the dependencies between containers were handled seamlessly, further enhancing the efficiency and reliability of the deployment process.

IV. Navigation

Another substantial enhancement involved the upgrade to the ROS2 Nav2 package [5]. A thorough understanding of the set of tools provided by the package and fine-tuning its many parameters was a point of focus within the team.

Our solution currently uses the slam_toolbox package [6] for mapping relying on odometer data from the Pioneer 3-DX and planar distance scans from the Hokuyo LiDAR. During this process, an image representing the map with a resolution of 5cm x 5cm is produced. In general navigation (i.e. without mapping), we use the Nav2 AMCL implementation for estimating the robot's position within the map. The remaining parts of the Nav2 stack are then responsible for calculating the best-cost trajectory considering the pre-loaded map and dynamic obstacle avoidance.

A. ROSAria2

In the update from ROS1 to ROS2, mainly motivated by recent support and the new features of packages such as Nav2, a problem was encountered. ROSAria, the application used for communication between the computer and the Pioneer 3-DX robot (Antares' platform), had not been ported to ROS2. As this is the main component of the robot's locomotion hardware, it is necessary to develop a solution for the data transmission between the interfaces provided by ROS2 and the serial communication of the locomotion base. To the best of the knowledge of the Warthog Robotics @Home group, there is no currently public solution available. Therefore, it was decided to create a new ROS package, zodiac_rosaria2, which plays the role of updating the ROSAria package to use ROS2.

Fig. 5. Object detection test using YOLOv8.

V. Computer Vision

Another major improvement was in the implementation of new computer vision capabilities. This year, the YOLOv8 [7] model was tested for object detection, offering a faster and more accurate way to identify items in the environment. The YOLOv8 model, with its advanced detection capabilities, has significantly improved the robot's ability to recognize and interact with household objects in dynamic environments. Additionally, the face_recognition package [8], [9] in Python was integrated for both face detection and recognition, enhancing the robot's ability to identify and interact with specific individuals. The face_recognition Python library, built upon the dlib [10] library, utilizes deep learning techniques for efficient face recognition. It employs a pre-trained convolutional neural network (CNN) to extract 128-dimensional embeddings from facial tex2025/images, effectively capturing unique facial features. These embeddings are then compared using Euclidean distance to determine facial similarity, enabling tasks such as face detection, recognition, and verification with high accuracy.

In addition, we integrated a feature that takes a frame from the Kinect V2 [11] sensor (used as the robot's vision system) and runs the YOLO model on it. When a person is detected, we send the center of mass of the detection to a skill that calculates the depth of this point using the Kinect's depth sensor. After that, we convert this point into a real-world coordinate using the Kinect's field of view (FOV) [12] and performing additional calculations and transformations. This process enabled us to develop a tracking skill, which allows the robot to follow a person (or point) autonomously.

VI. Human-Robot Interaction

For the human-robot interaction, we opted to use large language models (LLMs) to process the information received by the robot, as these models represent the state-of-the-art in several natural language processing tasks. Therefore, to select the models we chose only open-source models and evaluated

their performance considering their output and computational cost. The chosen models were RoBERTa [13] and LLaMA3.1-8B [14], which work well on quiz skills and can run locally. Our current work-in-progress implementation involves the application of these models for information extraction and dialogue text synthesis.

Likewise, we needed a speech recognition and a speech synthesis model. Therefore we wanted some models that could also run locally because depending on internet connection is not a good procedure for competition and household environments. Hence, we use Nix-TTS [15] for speech synthesis and locally run Whisper [16] for speech recognition. However, we still run Google's API for Speech-to-Text (STT) online when applicable.

VII. Conclusion

In this paper, we presented the development of our team for RoboCup @Home Brazil Open 2025. It is clear our effort and competence to perform the tasks in the competition despite our recent change in software architecture and solutions approach. For future work, we seek to keep evolving in the league and development of domestic robots, focusing on human-robot interaction and object manipulation, as we expect to scientifically contribute to the community and deliver an assistive robot capable of outstandingly helping disabled people with their routine and other activities.

References

[1] FreeRTOS. FreeRTOS: Real-time operating system for microcontrollers. 2024. Available at: https://www.freertos.org. Accessed on November 24th, 2024.

[2] Macenski, Steven, Foote, Tully, Gerkey, Brian, Lalancette, Chris, and Woodall, William. Robot Operating System 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66): eabm6074, 2022. doi: 10.1126/scirobotics.abm6074.

[3] Coleman, David, Sucan, Ioan, Chitta, Sachin, and Correll, Nikolaus. Reducing the barrier to entry of complex robotic software: a moveit! case study. *arXiv preprint* arXiv:1404.3785, 2014.

[4] Merkel, Dirk. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal*, 2014(239): 2, 2014.

[5] Macenski, Steve, Martín, Francisco, White, Ruffin, and Clavero, Jonatan Ginés. The Marathon 2: A Navigation System. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020. Available at: https://github.com/ros-planning/navigation2. Preprint: https://arxiv.org/abs/2003.00368.

[6] Macenski, Steve, and Jambrecic, Ivona. SLAM Toolbox: SLAM for the dynamic world. *Journal of Open Source Software*, 6(61): 2783, 2021. doi: 10.21105/joss.02783.

[7] Jocher, Glenn, Chaurasia, Ayush, and Qiu, Jing. Ultralytics YOLOv8, Version 8.0.0, 2023. Available at: https://github.com/ultralytics/ultralytics. License: AGPL-3.0.

[8] Geitgey, Adam. Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning. 2016. Published on Medium. Available at: https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78. Accessed on November 24th, 2024.

[9] Geitgey, Adam. face_recognition: The world's simplest facial recognition API for Python and the command line, 2018. Available at: https://github.com/ageitgey/face_recognition.

[10] King, Davis E. Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research*, 10: 1755–1758, 2009.

[11] Jamhoury, Lisa. Understanding Kinect V2 Joints and Coordinate System. 2018. Published on Medium. Available at: https://lisajamhoury.medium.com/understanding-kinect-v2-joints-and-coordinate-system-4f4b90b9df16. Accessed on November 24th, 2024.

[12] Smeenk, Roland. Kinect V1 and Kinect V2 fields of view compared. 2014. Available at: https://smeenk.com/kinect-field-of-view-comparison/. Accessed on November 24th, 2024.

[13] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint* arXiv:1907.11692, 2019.

[14] Dubey, Abhimanyu, Jauhri, Abhinav, Pandey, Abhinav, Kadian, Abhishek, Al-Dahle, Ahmad, Letman, Aiesha, Mathur, Akhil, Schelten, Alan, Yang, Amy, Fan, Angela, et al. The LLaMA 3 herd of models. *arXiv preprint* arXiv:2407.21783, 2024.

[15] Chevi, Rendi, Prasojo, Radityo Eko, Aji, Alham Fikri, Tjandra, Andros, and Sakti, Sakriani. Nix-TTS: Lightweight and end-to-end text-to-speech via module-wise distillation. In *2022 IEEE Spoken Language Technology Workshop (SLT)*, pages 970–976. IEEE, 2023.

[16] Radford, Alec, Kim, Jong Wook, Xu, Tao, Brockman, Greg, McLeavey, Christine, and Sutskever, Ilya. Robust speech recognition via large-scale weak supervision. In *International Conference on Machine Learning*, pages 28492–28518. PMLR, 2023.