# AURIS Robotic System – System Architecture

## 1. Operational Software Architecture

The AURIS robot software is organized in layered architecture, ensuring modularity, safety, and scalability.

1.1 Application Layer (AURIS Behaviors)
- Autonomous domestic tasks (pick, organize, transport objects).
- Proactive social interaction (voice + context awareness).
- Emergency handling and basic triage (detection and alert protocol).
- Transparency via VizBox interface (perception, planning, actions).

1.2 Decision Layer (Reasoning & Planning)
- Contextual home state representation.
- Task planner with user confirmation logic.
- Safety policies (speed limits, restricted zones, emergency priority).

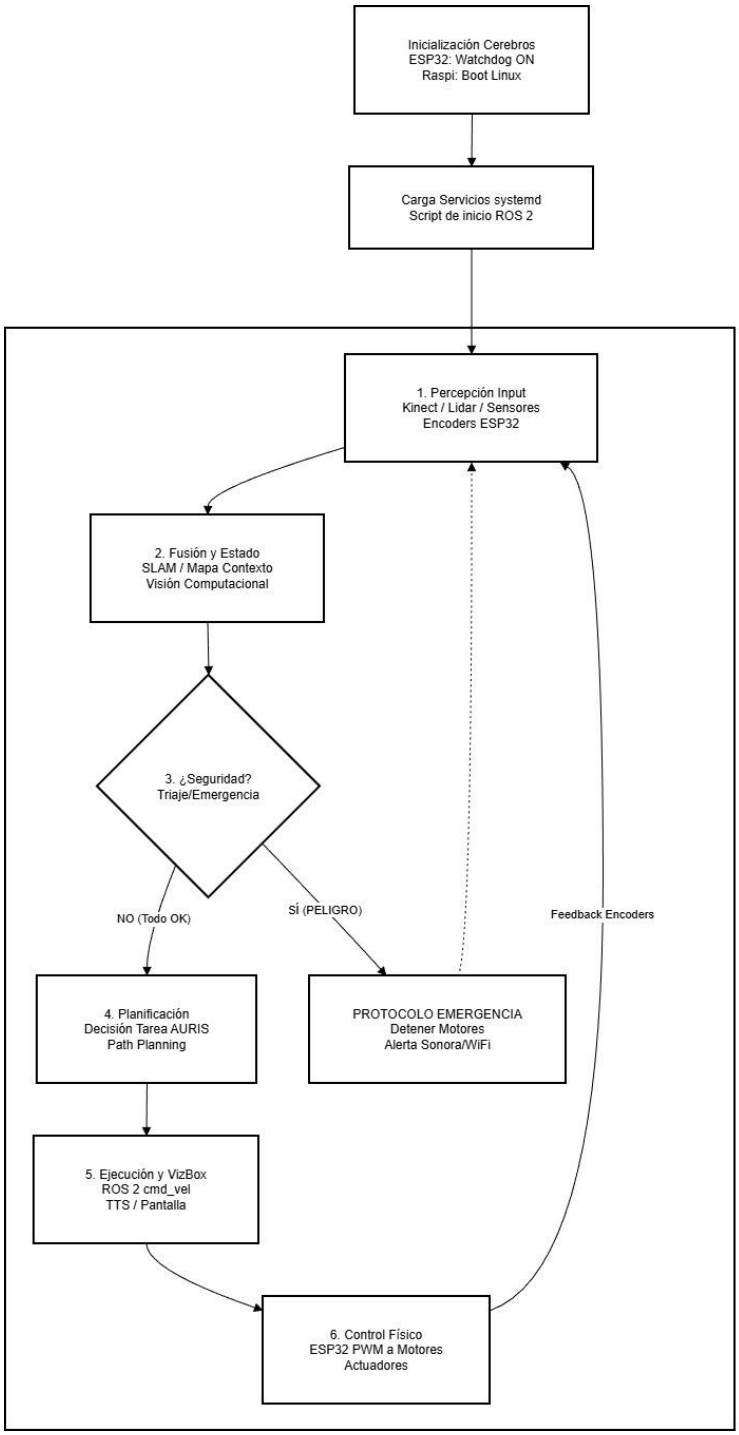
1.3 ROS 2 Middleware Layer
- Modular communication between perception, navigation, manipulation, interaction, safety, and monitoring.

1.4 Device Layer (Drivers)
- Sensor and actuator drivers.
- Real-time bridge with ESP32 for PWM, encoders, and emergency handling.

1.5 Operating System Layer
- Linux-based OS on Raspberry Pi 4.
- ROS 2 with Python nodes and systemd startup services.

## 2. Hardware Architecture

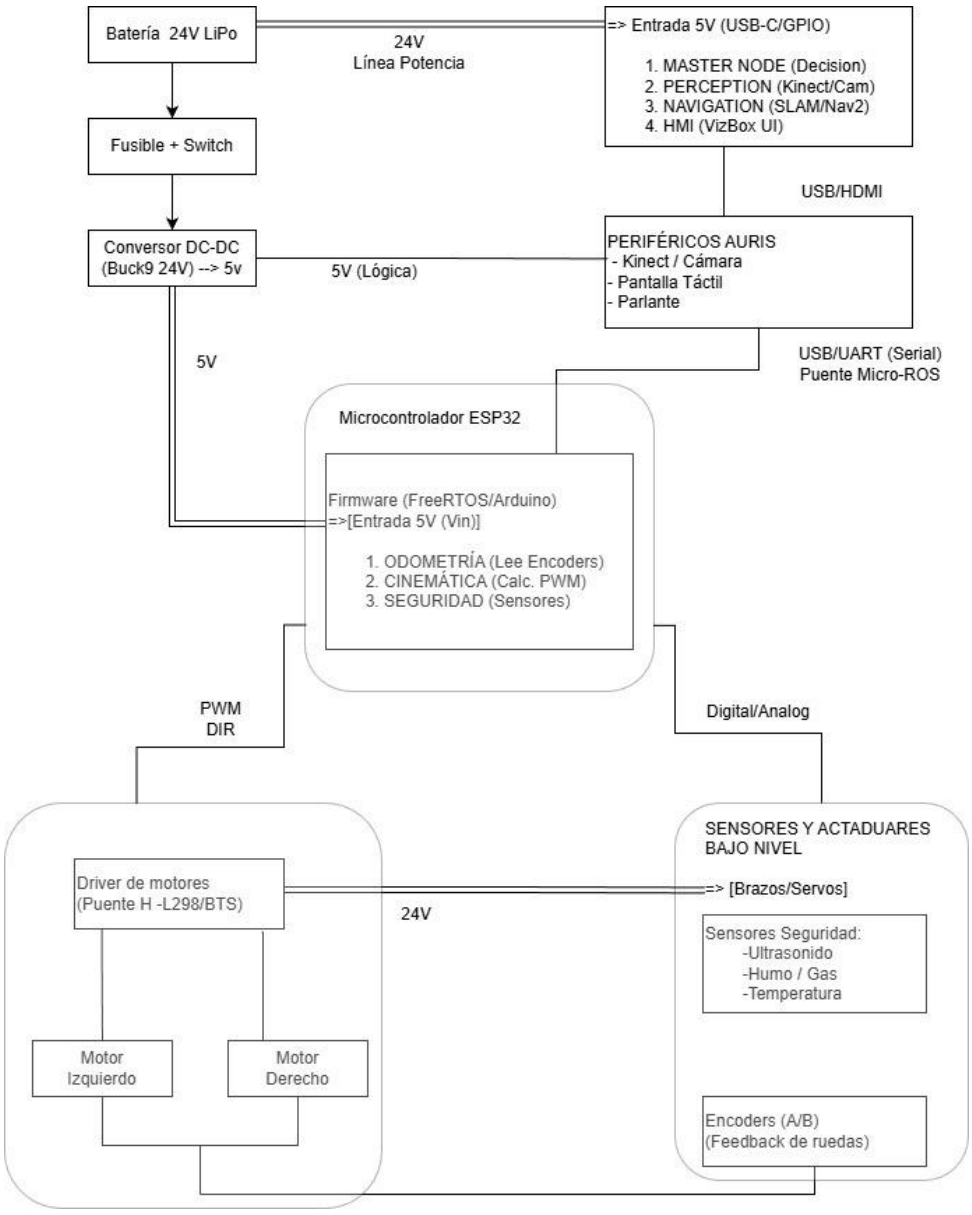
- Raspberry Pi 4 as main computational unit.
- ESP32 as real-time controller.
- Vision peripherals: Kinect, cameras, display, speaker.
- Safety and environment sensors (smoke, temperature, proximity, ultrasonic).
- Optional biomedical sensors.




First Picture: The figure illustrates the functional flow of the AURIS system. The system begins with multimodal perception (vision, sensors, and encoders), followed by information fusion and safety assessment. Under normal conditions, the system proceeds with task planning and execution using ROS 2, while in risk situations, an emergency protocol is activated. Physical control is delegated to the ESP32, which provides continuous feedback to the decision-making system.

## 3. Electrical Architecture (Block Explanation)

- Power flow: 24V Battery → DC-DC Buck → 5V Logic.
- ESP32 handles PWM motor control, encoders, and sensors.
- Motor driver controls two DC motors.
- Encoder feedback closes the control loop.




Second picture: The figure shows the electrical architecture of the AURIS system. The main 24V power supply is regulated by a DC-DC converter to 5V for the control electronics. The ESP32 microcontroller acts as a real-time control node, managing PWM signals to the motor driver, reading encoders for odometry, and monitoring safety sensors. The system implements a closed feedback loop for chassis movement control.

## 4. Materials and Components

- Aluminum T-slot mechanical structure.
- DC motors, encoders, wheels with brushes.
- Servomotors for arms and grippers.
- Raspberry Pi 4, ESP32, display, Kinect, cameras.
- Emergency button and safety sensors.

## 5. Diagram Representation Guidelines

- Block diagram for conceptual understanding.
- Wiring diagram for technical implementation.
- Connector tables including battery, drivers, motors, encoders, and ESP32 pins.