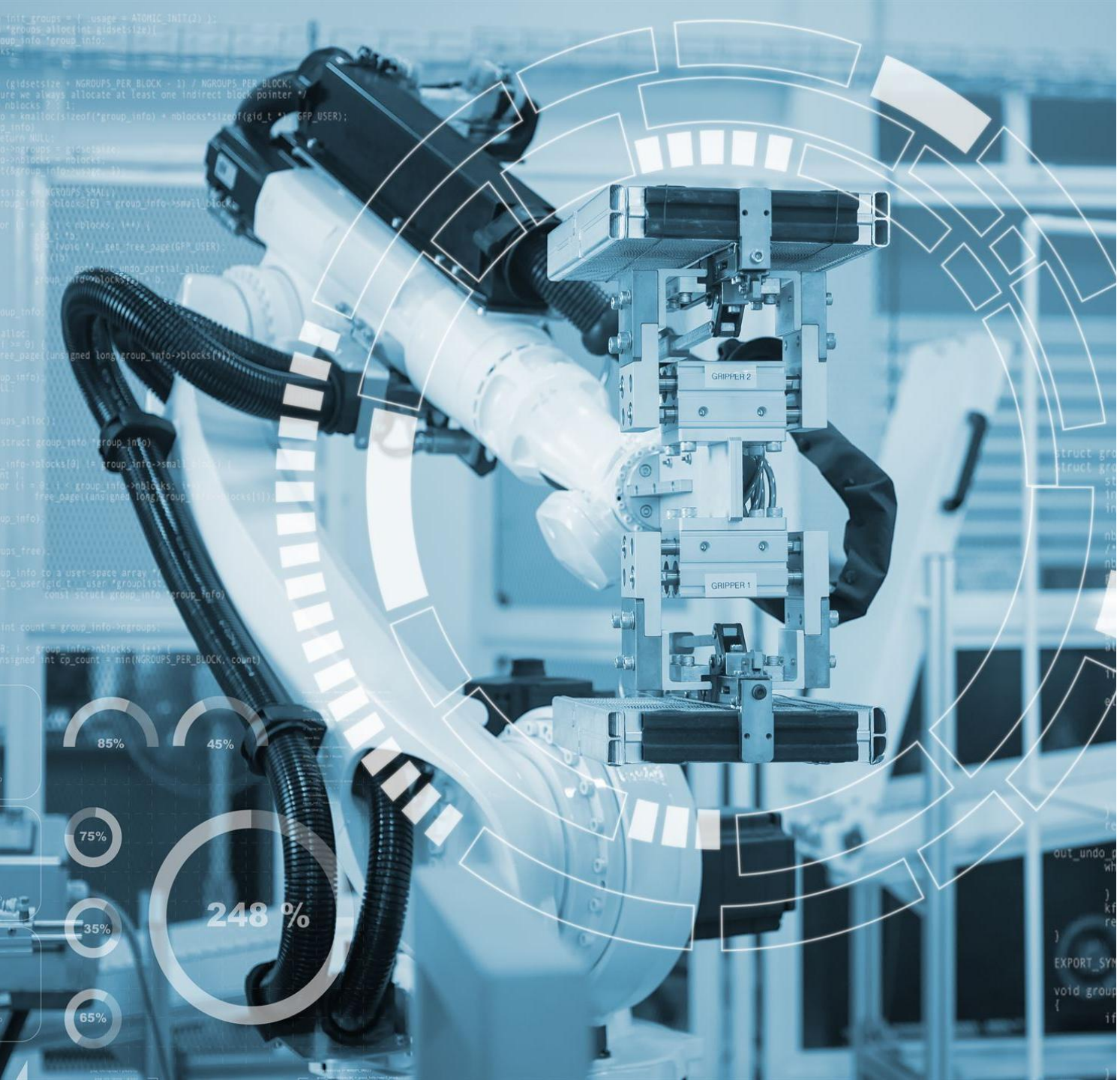


Applied AI & Robotics Portfolio

Showcasing practical innovations in intelligent robotics.

Andres Daza

Rodrigo Sierra

Professor Sitaram Ayyagari

ITAI 2374

Introduction

Portfolio Overview

Technical Achievements

The portfolio highlights practical implementations in edge AI, computer vision, and autonomous systems using advanced technologies and frameworks.

Ethical Considerations

A strong focus on building safe, ethical, and efficient robotic applications underscores the portfolio's responsible approach to AI deployment.

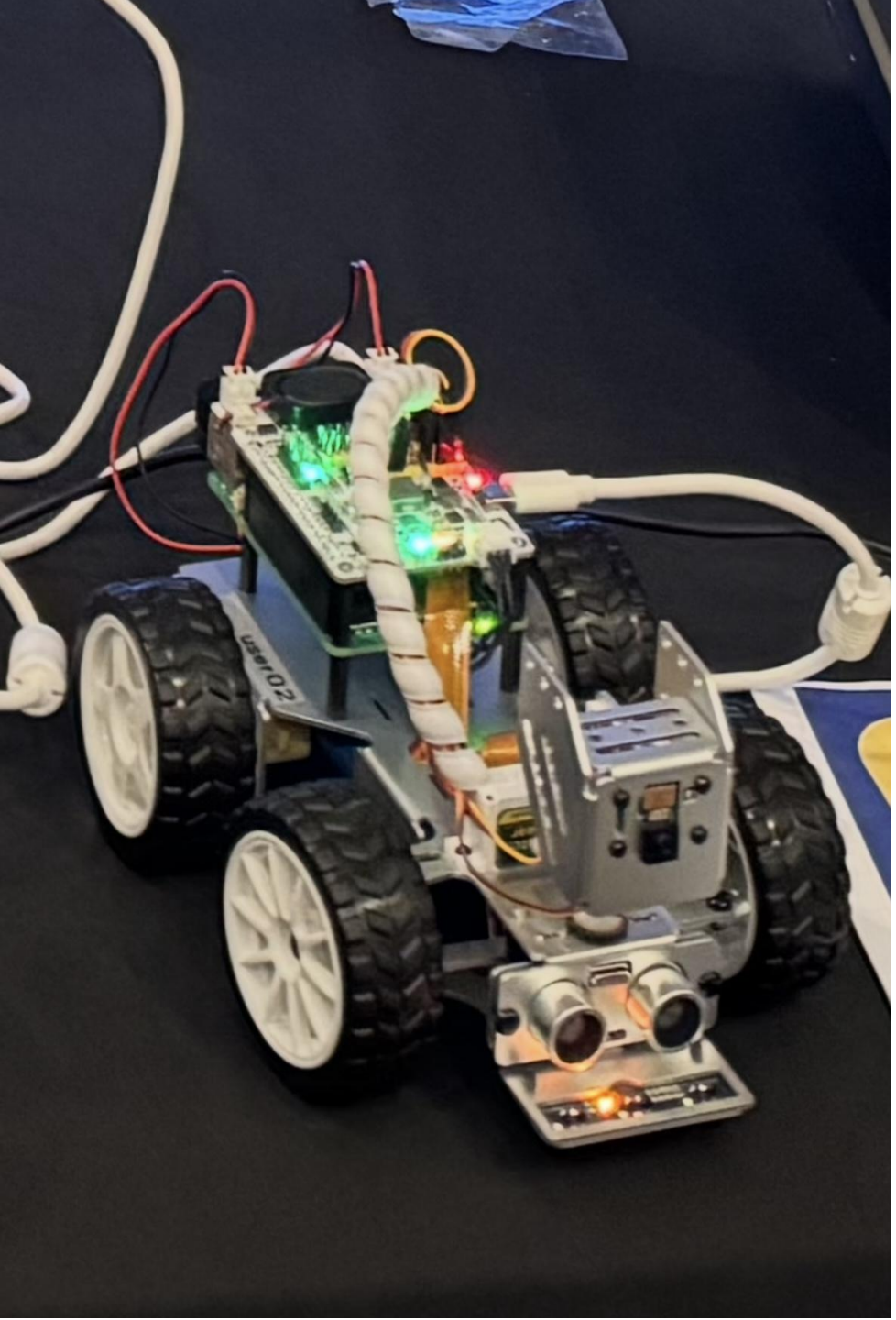
Technical Stack

The portfolio leverages a robust technology stack including Python, PyTorch, and Raspberry Pi for development and implementation.

Student's Approach

The portfolio provides a comprehensive view of the student's capabilities and approach to solving applied AI and robotics challenges effectively.

Project 1: Hardware Setup



Picar X Assembly & Configuration

This project focuses on the initial hardware assembly, firmware configuration, and software environment setup for the SunFounder Picar X (v2.0). The goal was to establish a functional mobile robotics platform for future computer vision and autonomous driving tasks.

Hardware & Tools Used

- **Robot Kit:** SunFounder Picar X (v2.0)
- **Controller:** Raspberry Pi 5
- **OS:** Raspberry Pi OS (Legacy/Bullseye recommended for camera compatibility)
- **Language:** Python 3

Implementation Steps

Assembly: Constructed the chassis, servo motors, and camera module following the mechanical schematics.

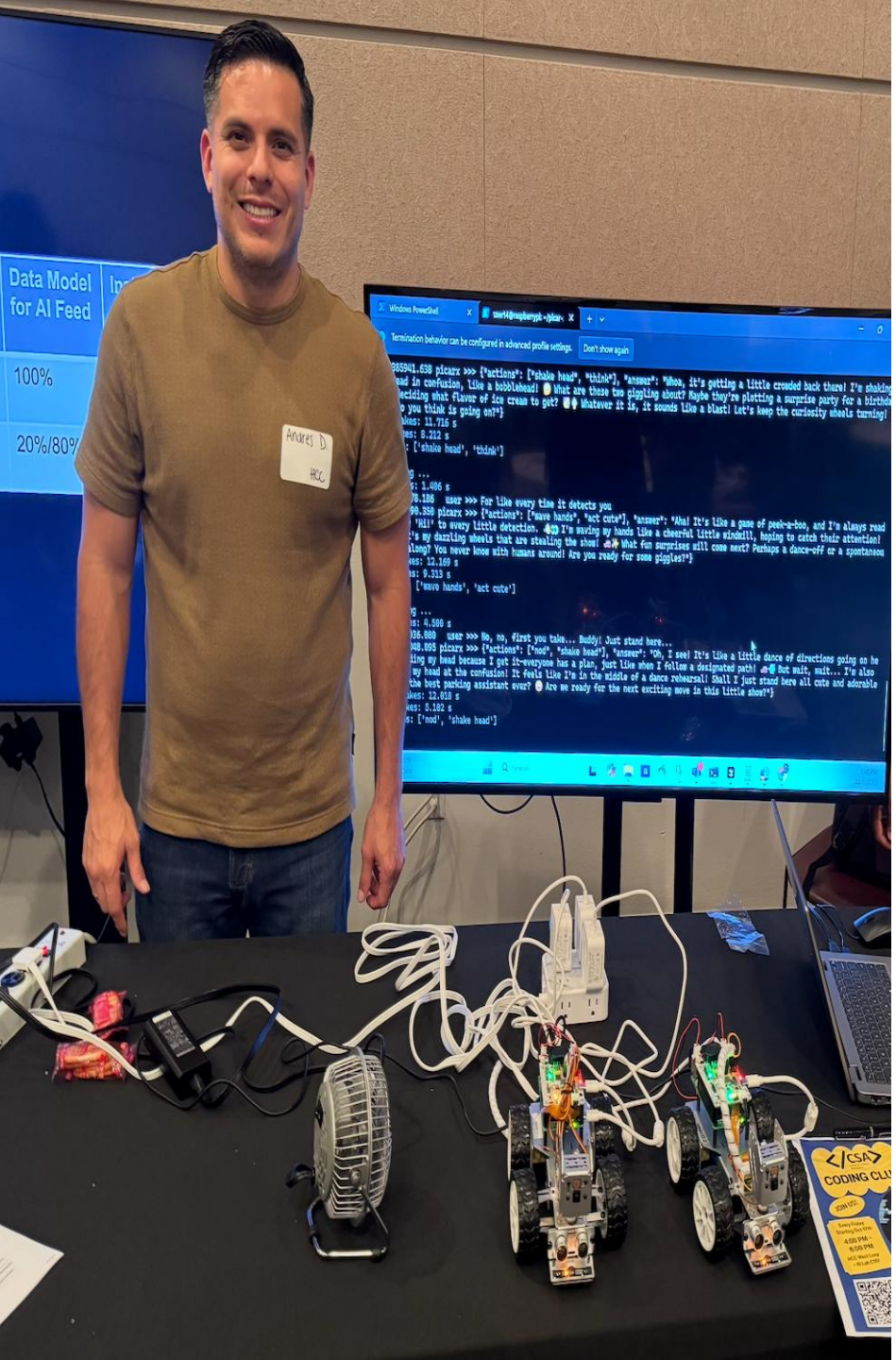
Environment Configuration:

- Enabled I2C, SPI, and Camera interfaces via raspi-config.
- Installed required dependencies (Python or robot_hat libraries).

Calibration:

- Performed servo zeroing to ensure wheel alignment.
- Calibrated the grayscale sensor for line tracking capabilities.

Project Final: Autonomous Patrol



Line Tracking with Emergency Stop

Project Goal

To design a functional robotic application that demonstrates sensor fusion, autonomous navigation, and safety protocols using the Picar X platform.

System Architecture

Sensors (Perception):

- 3-Channel Grayscale Module (Line detection)
- Ultrasonic Sensor (Time-of-flight distance measurement)

Actuators:

- DC Gear Motors (Propulsion)
- Servo Motor (Ackermann Steering)

Control Loop: Python-based generic decision loop running at ~50Hz.

Meeting the Requirements

1. Integration of Sensors & Actuators

The system fuses data from the grayscale module (floor contrast) and ultrasonic sensor (environment depth) to modulate the voltage to the DC motors and the angle of the steering servo.

2. Autonomous Navigation

The robot utilizes a logic-based control algorithm to autonomously track a high-contrast trajectory (black line) without external remote control or human intervention.

3. Real-Time Perception

The control loop continually reads sensor values. The steering angle is adjusted dynamically based on lateral error (deviation from the line center) in real-time.

4. Safe Operation

Safety Implementation: The code includes a priority interrupt for obstacle detection.

Logic: If distance < 15cm THEN Emergency Stop.

How to Run

sudo python3 patrol.py

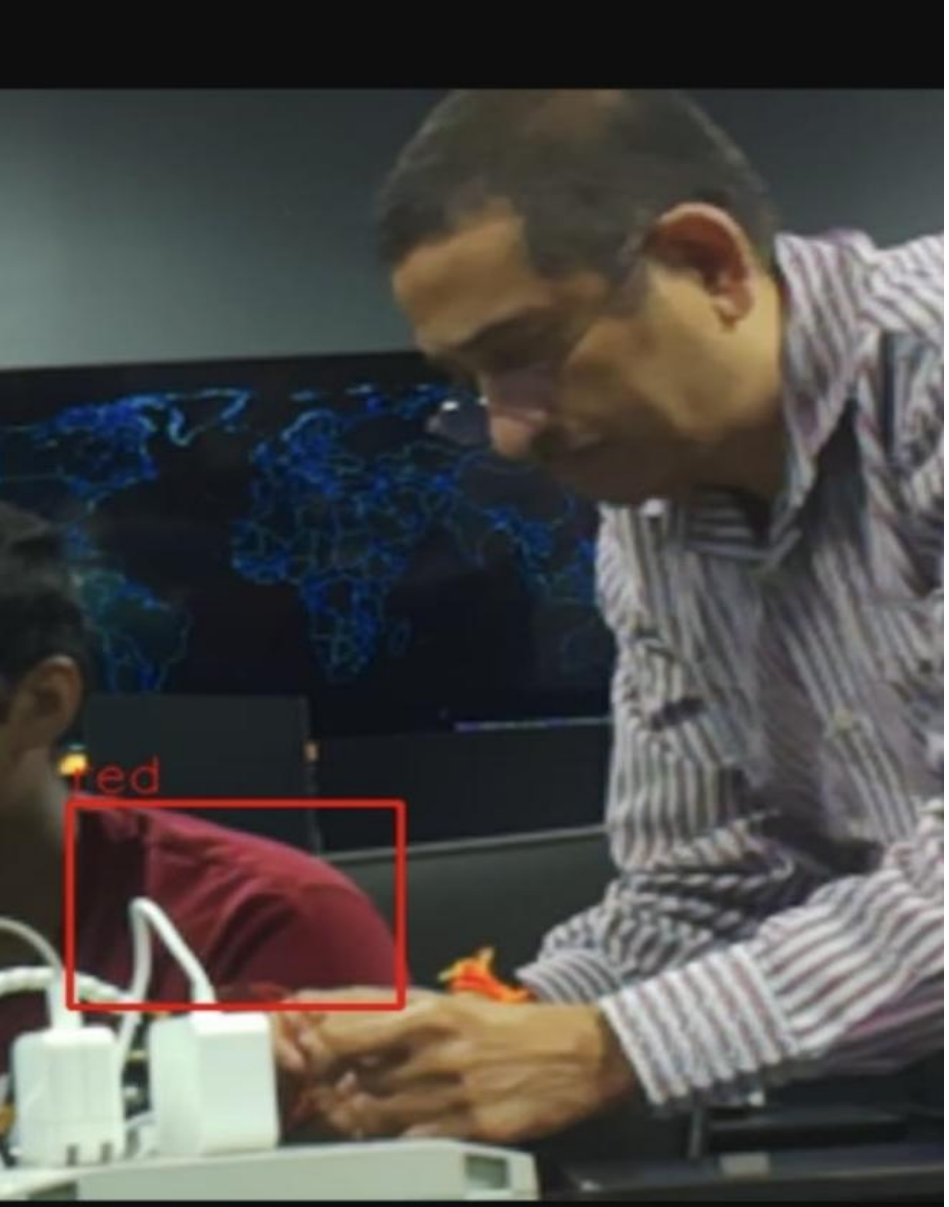
(Sudo is often required for GPIO access on Pi)

patrol.py

```python
from picarx import Picarx
from time import sleep

# Initialize the Robot
px = Picarx()

# --- CONFIGURATION ---
POWER = 20           # Motor speed (0-100) - Keep low for safety
SAFE_DISTANCE = 15   # centimeters
GM_THRESHOLD = 500   # Grayscale threshold (Adjust based on lighting)

def get_grayscale_status():
    """
    Reads the 3 grayscale sensors to determine where the line is.
    Returns: 'left', 'right', 'straight', or 'lost'
    """
    gm_val_list = px.get_grayscale_data()
    # gm_val_list looks like [left_sensor, center_sensor, right_sensor]

    # Logic: If sensor value is LOW, it sees Black (Line). High = White (F
    # Note: Adjust logic depending on if your line is black or white.
    # Assuming Black Line on White Floor:
    left = gm_val_list[0] < GM_THRESHOLD
    center = gm_val_list[1] < GM_THRESHOLD
    right = gm_val_list[2] < GM_THRESHOLD

    if center:
        return 'straight'
    elif left:
        return 'left'
    elif right:
        return 'right'
    else:
        return 'lost'
```

```python
def main():
    print("🤖 Autonomous Patrol Started")
    print("Press Ctrl+C to stop manually.")

    try:
        while True:
            # 1. PERCEPTION & SAFETY (Ethical Operation)
            if not safety_check():
                px.stop()
                print("⚠️ OBSTACLE DETECTED! Emergency Stop.")
                sleep(0.1)
                continue # Skip the rest of the loop, wait for path to clear

            # 2. NAVIGATION LOGIC (Real-time decision making)
            status = get_grayscale_status()

            if status == 'straight':
                px.set_dir_servo_angle(0)     # Wheels straight
                px.forward(POWER)
            elif status == 'left':
                px.set_dir_servo_angle(-35)   # Turn wheels left
                px.forward(POWER)
            elif status == 'right':
                px.set_dir_servo_angle(35)    # Turn wheels right
                px.forward(POWER)
            elif status == 'lost':
                # Safety behavior: If line is lost, stop to prevent running away
                px.stop()

            sleep(0.01) # Small delay for stability

    except KeyboardInterrupt:
        print("\n🛑 User Interrupt. Stopping.")
    finally:
        px.stop()

if __name__ == "__main__":
    main()
```

Ethical Reflection & Lessons



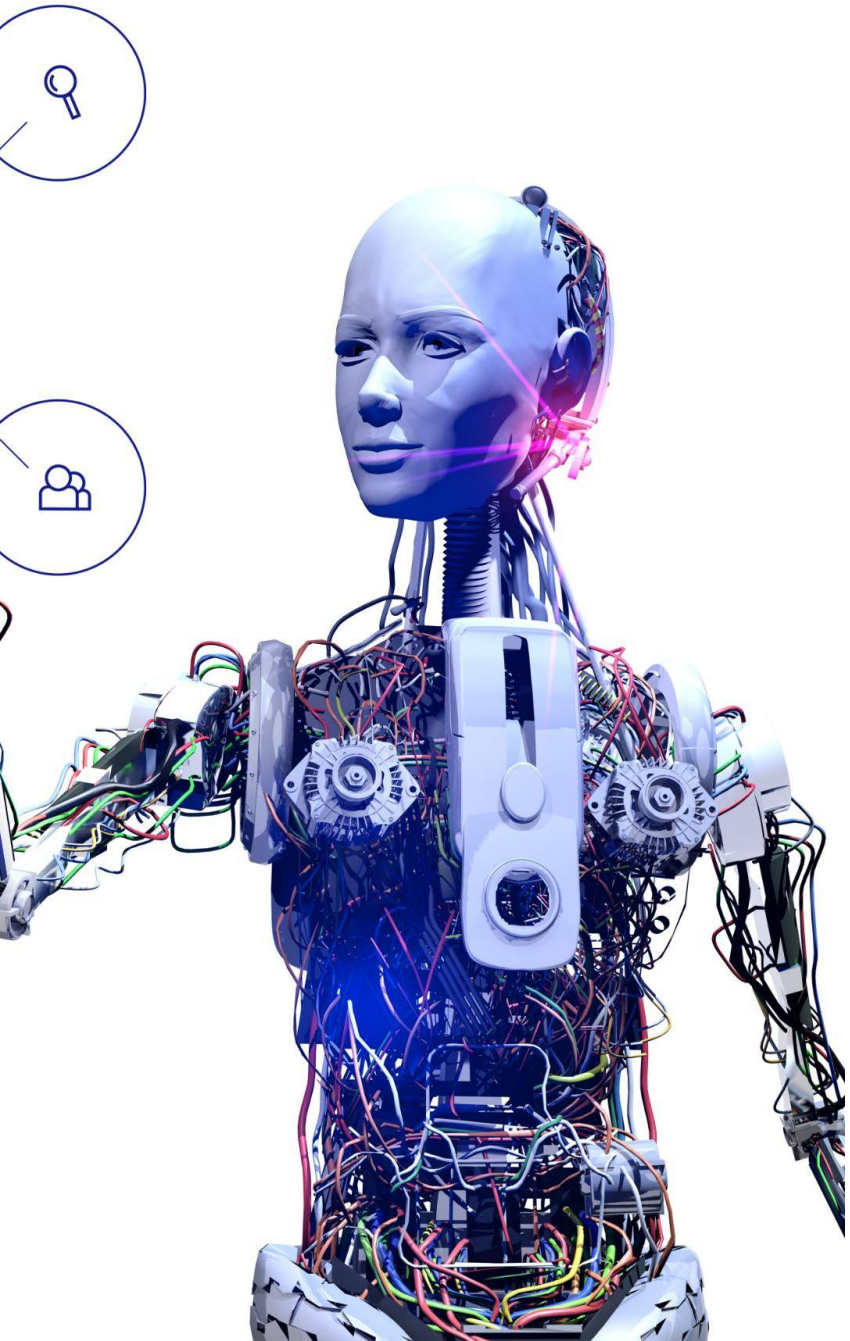
Ethical & Technical Reflection: Autonomous Mobile Robotics

Responsible Decision Making

In the design of the autonomous Picar X application, responsible decision-making was encoded directly into the control loop's hierarchy. The robot was programmed to prioritize "safety" over "mission completion." Specifically, the `safety_check()` function executes before any navigation logic. By reading the Ultrasonic sensor data first, the system ensures that if an obstacle is detected within 15cm, the decision to stop overrides the decision to follow the line. This demonstrates a foundational principle of responsible robotics: a system must be aware of its environment and possess a "fail-safe" state to prevent harm to itself or its surroundings.

Safety and Privacy Considerations

Safety was the primary operational concern. The physical design of the robot is small and lightweight, but kinetic energy can still cause damage. To mitigate this, the motor power was capped at 20% duty cycle, ensuring low speeds that allow for reaction time. Regarding privacy, this specific design is inherently privacy-preserving. Unlike systems that rely on cloud-based LLMs or facial recognition cameras, this robot processes data locally using simple infrared (grayscale) and ultrasonic sensors. It does not record, store, or transmit personal data or visual feeds, effectively eliminating the risk of surveillance or data leakage.



Ethical & Technical Reflection: Autonomous Mobile Robotics

Improving Accessibility and Inclusivity

Currently, the robot communicates its state only through movement; if it stops, a user might not know if it is broken, paused, or detecting an obstacle. To improve accessibility, specifically for visually impaired users, I would integrate audio feedback (using a buzzer or text-to-speech module) to announce "Obstacle Detected" or "Path Lost." For users with hearing impairments, integrating the RGB LED module to flash red for "Stop" and green for "Go" would provide necessary visual cues. Making the robot's internal state "transparent" through multi-modal feedback (sound and light) is crucial for inclusive design.

Lessons on Teamwork and Trust in AI

Developing this system reinforced that trust in AI is built on predictability, not just intelligence. We learned that for a human to trust a robot, the robot's behavior must be consistent. During testing, trust was established only after repeated trials proved that the emergency braking logic worked 100% of the time. Furthermore, this project highlighted the "human-robot team" dynamic. The robot is not truly fully autonomous; it relies on the human to define the parameters (the safe distance, the speed) and the environment (the track). True success requires the developer to anticipate failure modes, proving that an AI is only as reliable as the ethical frameworks and safety constraints defined by its human creators.

Thank You,

View the full code at:
\github.com/AndresDazaTech/Robotics-Portfolio
\github.com/RodMLGH/Rodrigo-Sierra-Robotics-Portfolio