# Introduction

This project addresses autonomous robot navigation using RISC-V assembly. The primary motivation for selecting this topic is the growing significance of embedded systems in robotics, particularly navigation tasks requiring efficient sensor data processing and motor control. The relevance of the project lies in its application to robotics systems deployed in hazardous or inaccessible environments.

# Methodology

The core of our implementation is a navigation algorithm written entirely in RISC-V assembly. The robot navigates by processing inputs from infrared (IR) and ultrasonic sensors. Essential RISC-V features such as registers (e.g., t0–t6, s0–s3), conditional branching (beq, bne, blt), and memory management (lw, sw) were utilized.

The algorithm follows these steps:

1. Initialize sensor input ports.
2. Continuously read sensor values.
3. Interpret sensor data for obstacle detection.
4. Execute motor control signals for navigation (forward, turn left/right, stop).

## Code Snippet: Sensor Initialization

li t0, 0xFFFF # Load mask to read full sensor input

lw t1, 0x10010000 # Load sensor data (IR/ultrasonic)

and t2, t1, t0 # Mask sensor input to isolate relevant bits

*Explanation:* This segment reads the raw sensor data from memory and masks it to extract only the relevant bits. These bits represent the presence of obstacles.

## Code Snippet: Navigation Decision Logic

beq t2, zero, forward # If no obstacle detected, go forward

bne t2, zero, turn # If obstacle detected, turn

*Explanation:* Based on the value in t2, the robot decides whether to move forward or perform a turning maneuver to avoid an obstacle.

## Code Snippet: Forward Movement

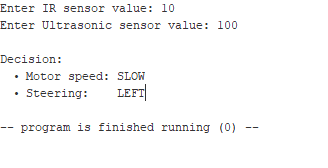
li a0, 1 # Code for 'move forward'

ecall # Simulate motor control system call

*Explanation:* This snippet issues a command to the robot to proceed forward. The actual motor control is simulated using ecall in the RARS environment.

# Results

The assembly code execution was simulated using the RARS environment. The results were validated by observing real-time register changes, sensor input handling, and motor output commands.



# Challenges & Solutions

* **Alignment and Memory Issues:** Addressed by ensuring word-aligned memory access.
* **Sensor Input Reading:** Optimized by efficient looping and minimal branching instructions.
* **Debugging Motor Behavior:** Manual validation via register tracking in RARS was required to ensure the robot turned in the correct direction.

# Conclusion

Throughout the project, we gained insights into real-time embedded programming challenges, RISC-V instruction optimization, and sensor data processing. Future improvements include implementing more complex navigation algorithms (path planning and avoidance strategies) and integrating additional sensor types for improved robustness.