YZV406E Robotics Project Proposal

Fall 2025–2026 Term

Istanbul Technical University

Group Members

Name	Student ID	Email
Eren Onaran	150210114	onarane21@itu.edu.tr
Hakan Çetinkaya	150200114	cetinkayah20@itu.edu.tr
Osmancan Sarı	150230728	sario20@itu.edu.tr

Project Topic

Title: Autonomous Cafeteria Delivery Robot

The ITU MED Cafeteria often experiences high congestion at the order counter, where students must wait for their food to be prepared. This creates a bottleneck and an inefficient experience. We propose the "Autonomous Cafeteria Delivery Robot", a mobile service robot designed to deliver food directly from the counter to the student's table. This project will deliver a complete "pick-up-free" service.

The cafeteria environment is highly dynamic: people frequently walk between tables, sometimes move tables away from their original positions, and may take the meal before the robot arrives or not take it at all. The robot may also be accidentally bumped by students, which could risk spilling food. In addition, lighting conditions can vary, with some areas becoming dim during busy periods or in the evening. These factors significantly influence the design requirements of our delivery robot, which require robust perception, reliable navigation, and well-defined contingency plans to handle unexpected human behaviors and environmental changes.

1. Description

- **Home:** The TurtleBot3 waits at a "Home" station by the counter.
- **Loading:** An employee places the food order onto the robot's tray and sends a command via the interface.
- **Navigation in Dynamic Environment:** The robot autonomously navigates to the specified table. Unlike standard static navigation, the robot utilizes a local planner configured to detect with LIDAR sensor and react to dynamic obstacles

(simulated walking students/actors). It replans its path to avoid collisions with moving objects.

- **Collision Handling & Operator Intervention:** The robot uses onboard sensors (IMU/Bumpers) to detect collisions. If a collision occurs:
  - The robot immediately performs an Emergency Stop.
  - It sends a signal to the interface at counter.
  - A human operator connects to the robot's camera feed remotely to inspect the tray.
  - The operator sends a command: either "Resume" if food is safe or "Abort/Return" if food is spilled.
- **Theft Prevention:** If the food is removed mid-route, detected via a force-sensitive resistor (FSR) embedded in the tray, the robot halts and initiates a 30-second grace period instead of aborting the delivery immediately.
  - The robot records a timestamped snapshot at the moment the food is removed.
  - If the food is returned within the 30-second window, a notification is sent to the counter interface.
  - A human operator can then connect to the robot's camera feed remotely to inspect the tray.
  - The operator issues a command: "Resume" if the food is intact, or "Theft" if the tray is empty or the meal appears compromised.
  - If the timer expires with no return, or a "Theft" command is sent, the robot triggers an audible alarm to deter the offender and navigates back to Home.
- **Validation & Search:** Upon arrival, the robot scans for the table's visual marker.
- **Contingency (Table Moved):** If the table's fiducial marker is not detected at its expected coordinate (e.g., the table has been moved), the robot initiates a "Recovery Rotation" to scan the surrounding area. If the marker is still not detected within a predetermined interval, the robot notifies the table as displaced and returns to the Home position. The robot then suspends service to that table until a human operator updates its location.
- **Delivery & Timeout Logic:** The robot signals arrival and waits for the user to retrieve the food.
- **Contingency (No Pickup):** A timer is started upon arrival. If the food is not taken within 60 seconds (student absent), the robot logs the failure and switches to a "Return with Food" state.

- **Return:** Once the delivery is confirmed (or the timeout expires), the robot navigates safely back to the counter to await the next task.

This project will integrate three core robotics problems: autonomous navigation, environment perception, and high-level task planning.

Timeline

Week	Milestone
Week 0	Proposal Submission
Week 1	**Setup & Simulation Environment.** Create the Gazebo world (MED Cafeteria). Import TurtleBot3 model and configure dynamic actor plugins to simulate walking students.
Week 2-3	**Parallel Module Development.** *Nav:* TEB/DWA local planner tuning for dynamic obstacles. *Perception:* Marker detection, FSR (tray weight) and IMU collision logic implementation. *Planning:* Coding the complex State Machine (Theft, Emergency Stop, Recovery).
Week 4-5	**System Integration & Contingency Testing.** Combining modules. Specifically testing failure scenarios: Food removal (Theft), Table missing (Recovery Search), and Operator Intervention interface.
Week 6	**Final Testing & Video Recording.** The full scenario is run repeatedly including "Return with Food" and "Theft" cases. A final demo video is recorded.
Week 7	**Finalization**. Video editing (with voice-over/subtitles) and project submission.

Individual Tasks for Each Group Member

Member 1: Safe Navigation in Dynamic Environments (Hakan Çetinkaya)
Distinct Problem: How does the robot navigate safely among moving students and recover if the goal is unreachable?

- **Dynamic World Creation:** Design the MED Cafeteria in Gazebo. Crucially, integrate "actor" plugins to simulate students walking and moving obstacles to test dynamic avoidance.

- **Local Planner Tuning (TEB/DWA):** Configure the ROS Navigation Stack's local planner to handle dynamic obstacles. Optimize parameters to ensure the robot slows down or re-plans rather than freezing when a person crosses its path.
- **Recovery Behaviors:** Implement the "Recovery Rotation" logic within the navigation stack to allow the robot to spin and re-localize if the table marker is not immediately visible.
- **Deliverable:** A navigation subsystem that detects moving obstacles via LIDAR and successfully navigates to coordinates even when the path is partially blocked by moving actors.

Member 2: Perception & Sensor Integration (Eren Onaran)

Distinct Problem: How does the robot perceive the table, detect collisions, and know if food is stolen using onboard sensors?

- **Visual Marker Detection:** Develop the ROS node using OpenCV to detect table fiducial markers for the final approach validation.
- **Tray Sensor (FSR) Logic:** Simulate and integrate a Force Sensitive Resistor (FSR) on the tray. Write logic to publish a `/food_status` topic (Safe/Removed) to trigger the theft prevention routines.
- **Collision Detection (IMU/Bumper):** Create a node that monitors IMU accelerometers and bumper data. If a spike/impact is detected, publish a high-priority `/collision_alert` message.
- **Camera Stream Management:** Implement the logic to capture a timestamped snapshot when a theft attempt is detected and manage the video stream for the remote operator.
- **Deliverable:** A comprehensive perception driver that handles visual markers, weight sensing (FSR), and physical impact detection (IMU).

Member 3: Integration and High-Level State Machine (Osmancan Sarı)

Distinct Problem: How do we orchestrate complex behaviors like "Theft Prevention", "Operator Intervention", and "Recovery" in a unified system?

- **Advanced State Machine Design:** Architect the central brain of the robot. The logic must now handle interrupt-driven events (collisions, theft).
- **State Logic Implementation:**

  - `STATE_NAVIGATING`: Standard travel, but listening for `/collision_alert` or `/food_status` changes.
  - `STATE_THEFT_GRACE`: Triggered by FSR. halts robot, starts 30s timer, waits for food return or timeout.
  - `STATE_EMERGENCY_STOP`: Triggered by IMU/Bumper. Halts motors, requests Operator Intervention.
  - `STATE_RECOVERY_SEARCH`: If marker is missing, execute rotation pattern to find the table.
  - `STATE_RETURN_WITH_FOOD`: If user pickup timeout (60s) expires, navigate back home with the tray full.
- **Operator Interface Logic:** Define the ROS service calls that allow the human operator to inject commands (`Resume`, `Abort`, `Theft_Confirmed`) into the state machine.
- **Deliverable:** The master launch file and the Python-based Finite State Machine that integrates navigation and perception inputs to execute the full delivery and contingency scenarios.

References

1. ROS Wiki: *turtlebot3_navigation*, `https://wiki.ros.org/turtlebot3_navigation`
2. OpenCV Documentation, `https://docs.opencv.org/`
3. "Adaptive Monte Carlo Localization for Mobile Robots," Dieter Fox et al., AAAI 1999.
4. Cheong, A., Lau, M. W. S., Foo, E., Hedley, J., & Bo, J. W. (2016). Development of a robotic waiter system. IFAC-PapersOnLine, 49(21), 681-686.
5. Babu, A. V., MJ, A. K., Damodaran, S., James, R. K., Kumar, A. V., & Warrier, T. S. (2024, September). Enhancing Mobile Robot Navigation in TurtleBot3 Burger: A ROS-Enabled Approach Focusing on Obstacle Avoidance in Real-World Scenario. In 2024 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES) (pp. 1-6). IEEE.
6. Abbas, Z., Abbas, S., & Mazhar, A. (2019, March). Automatic Cafe Management System Using Waiter Robot. In 2019 2nd International Conference on Communication, Computing and Digital systems (C-CODE) (pp. 211-214). IEEE.
7. Nafis, A. B., Ahammed, K., Oishi, H. R., Afroj, S., & Raka, R. C. (2024). A comprehensive study and analysis of artificial intelligence-based waiter robot in restaurant (Doctoral dissertation, Brac University).