

Çukurova University

Computer Engineering

GoalPilot

Goal-Biased Autonomous Navigation
in an Unknown Environment

... : : : :



Table of Contents

1

Project Goal

Problem statement & objective

2

Problem Constraints

Requirements, limits,
success criteria

3

System Architecture

Block diagram &
components

4

ROS Nodes & Dataflow

Node graph & topics

Table of Contents

5

Algorithm Loop

Select-navigate-replan

6

Challenges & Solutions

Versioned fixes

7

Results

Capabilities & parameters



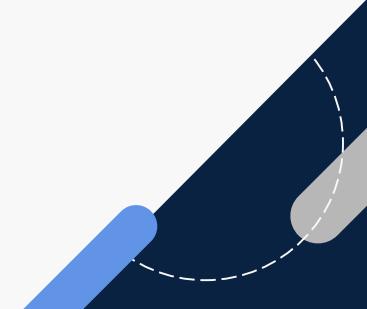
Project Goal

Problem statement

- Build a simulated mobile robot that can navigate autonomously in an unknown environment.
- The robot must not use a pre-saved map and must still explore, avoid obstacles, and plan paths to reach a target.

Objective

- Develop the system in ROS + Gazebo and visualize/debug in RViz.
- Apply ROS fundamentals (nodes, topics, services, parameters, launch files) to build a working navigation pipeline.
- Deliver a clear demonstration and explain the design, challenges, and results.



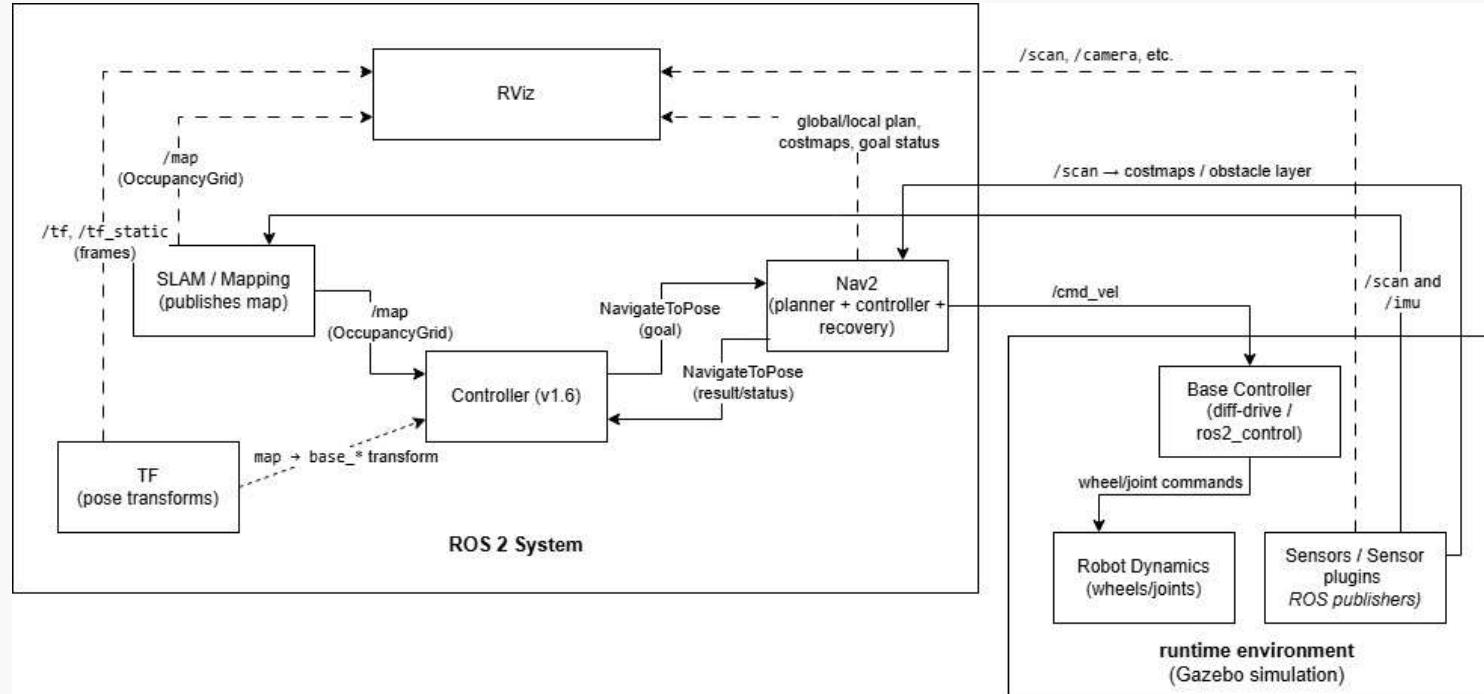
Problem Constraints

- **Unknown environment:** the map is not available at the start
- **No pre-saved map allowed:** mapping must be built online during the run
- Must support **exploration + navigation** in the same system
- Must handle **path planning** while moving toward the target
- Must perform **obstacle avoidance** to stay collision-free
- Implemented and tested in **ROS + Gazebo**, visualized in **RViz**

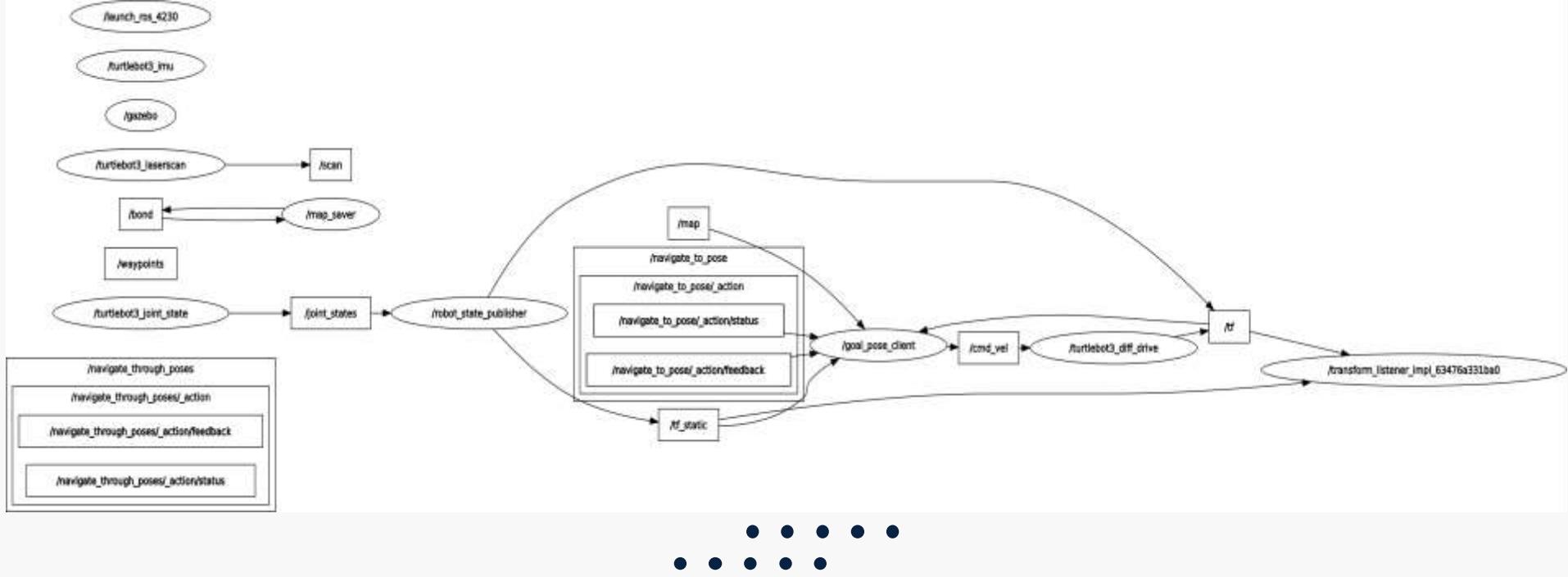
• • • • •
• • • • •



System Architecture



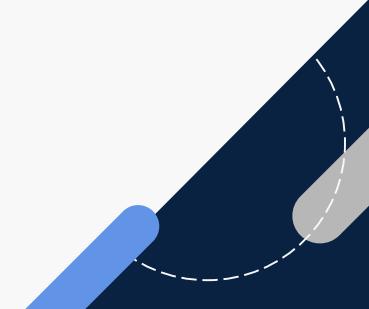
ROS Nodes & Dataflow





Algorithm Loop

- 1) **Wait for inputs:** /map available + TF map → base_* pose
- 2) **Target selection (each attempt):**
 - If main goal is known + safe → go directly
 - Else choose a subgoal near the goal (sample rings around goal, filter safe cells)
 - Score candidates: closer to goal + higher clearance + frontier bonus
- 3) **Execute:** send Nav2 NavigateToPose (yaw strategy: subgoals face goal; final goal keeps yaw to reduce spinning)



Algorithm Loop

- 4) **Monitor:** wait for SUCCEEDED / ABORTED, with a hard timeout (cancel if stalled)
- 5) **Replan:**
 - If SUCCEEDED at main goal → done
 - If SUCCEEDED at subgoal → enable FINAL MODE, try main goal again
 - If ABORTED/timeout → optional small backup+rotate if no progress, expand search radius, retry up to max attempts





Challenges & Solutions

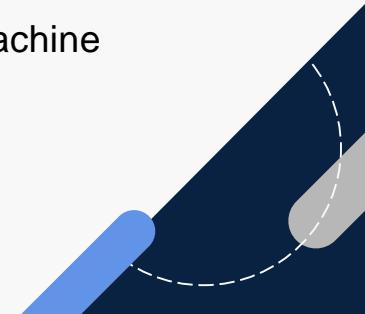
Iteration Timeline

v1.0 – Reactive exploration was unreliable

- **Challenge:** Random wandering often oscillated near walls and revisited the same space; goal stayed unknown.
- **Solution:** Baseline “explore until goal cell is known, then send Nav2 goal” (worked sometimes, not consistent).

v1.1 – Robot gets stuck / no progress

- **Challenge:** Robot could appear moving but made no real progress (local traps).
- **Solution:** Added odometry-based stuck detection + a small exploration state machine (forward / rotate / escape trigger).



Challenges & Solutions

Iteration Timeline

v1.2 - v1.3 – Escaping tight spaces and local minima

- **Challenge:** Simple rotate/forward behaviors still failed in corners and narrow passages.
- **Solution:** Implemented structured escape maneuvers (backup + angle sweep / probe) and early stop to re-check the goal periodically.

v1.4 – Major redesign: goal-biased exploration

- **Challenge:** “Explore first, then navigate” is inefficient and not goal-directed.
- **Solution:** Switched to map-driven intermediate targets (subgoals) near the goal:
 - world↔map utilities, safety dilation, frontier-like candidates,
 - TF-based pose in the map frame for structured replanning.





Challenges & Solutions

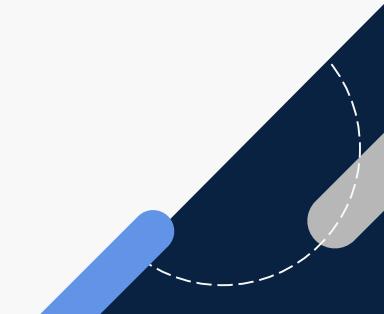
Iteration Timeline

v1.5 – Better subgoals + Nav2 edge-case handling

- **Challenge:** Some subgoals were technically valid but low quality; Nav2 behavior could stall or end awkwardly.
- **Solution:** Added candidate scoring (goal distance + clearance + frontier bonus), plus FINAL MODE to attempt the true goal when appropriate, and tighter Nav2 monitoring.

v1.6 (Final) – Stabilize and simplify

- **Challenge:** End-game oscillation and overly complex heuristics reduced consistency.
- **Solution:** Kept subgoal + scoring, but stabilized execution:
 - hard timeout + cancel if stalled,
 - simplified recovery (backup+rotate only when no movement),
 - yaw strategy to reduce spinning near the end.



Results

Validated Capabilities

- **Goal-biased navigation without a pre-saved map:** if the main goal is not safe/known, the robot selects map-based subgoals near the goal and keeps replanning.
- **Clear success condition:** run finishes only when Nav2 returns SUCCEEDED at the main goal.
- **More stable execution:** each Nav2 goal has a hard timeout; stalled goals are canceled and replanned.
- **Reduced end-game spinning:** subgoals face the main goal, but the final goal keeps current yaw by default.
- **Simple recovery only when needed:** if Nav2 aborts and the robot barely moved, apply backup + rotate, then retry.





Results

Key parameters

- **Safety / sampling:** SAFETY_CELLS=4, SEARCH_MAX_RADIUS=0.9 m, SEARCH_STEP=0.05 m
- **Scoring weights:** W_GOAL=1.0, W_CLEAR=1.8, W_FRONTIER=0.35, CLEARANCE_CAP_M=2.0
- **Robustness limits:** NAV2_GOAL_TIMEOUT_SEC=90 s, MAX_REPLANS=30
- **Yaw / recovery toggles:** SEND_FINAL_YAW_WITH_NAV2=False, ENABLE_SIMPLE_RECOVERY=True
- **Recovery motion:** BACKUP_DIST=0.20 m, ROTATE_DEG=25°

