HOCHSCHULE
ESSLINGEN

Application Project Planning
Application Project

Students:
Ricardo Sierra Roa
Sergi Fernández Mendez

Profesor:
Thao Dang

Hochschule Esslingen
Campus Hiltop

Fecha de entrega:
November 28th, 2025

Index

Project Overview

This project enables a TurtleBot4 Pro to follow a person smoothly and reliably using its onboard OAK-D Pro camera and ROS 2 Humble. Detection is powered by a YOLOv8 model trained in PyTorch and deployed to the camera via ONNX → OpenVINO blob, so inference runs on-device while the Raspberry Pi focuses on control and messaging.

Objectives

- Detect the person class in real time and obtain 3D position (X, Y, Z) from the OAK-D Pro.
- Track and select a single target consistently, even in crowded scenes.
- Generate smooth velocity commands to follow at a comfortable distance.
- Provide clear interfaces (topics, messages) for easy integration with navigation or logging tools.
- Keep CPU usage on the robot low by offloading inference to the camera NPU.

Hardware

- TurtleBot4 Pro (iRobot Create 3 base + Raspberry Pi 4)
- Luxonis OAK-D Pro (RGB + stereo depth + IR projector)
- Development laptop (for model training and testing)
- Power and networking accessories as required

Technology Stack

- ROS 2 Humble (Ubuntu 22.04) for middleware and lifecycle management.
- OAK-D Pro / DepthAI for spatial AI and depth estimation.
- YOLOv8 (Ultralytics, PyTorch) for model training and evaluation.
- ONNX + OpenVINO blob for portable, efficient on-device inference.
- Custom ROS 2 node (person_follower) for control and decision logic.

System Architecture (High-Level)

- Perception Layer: OAK-D Pro runs the compiled neural network and publishes spatial detections (bounding box + 3D coordinates + confidence).
- Tracking Layer: Maintains the selected target over time to avoid ID swaps.
- Control Layer: Computes linear and angular velocities from distance and heading errors, applying filtering, limits, and safety rules.
- Robot I/O: Velocity commands are sent to the Create 3; sensor data and status are exposed via standard ROS 2 topics.

Core Functionality

- Real-time person detection with depth-aware localization.
- Target selection and hand-off management when multiple people appear.
- Smooth motion with bounded velocities, rate limiting, deadbands, and safety stops.
- Clear, documented topics for detections, diagnostics, and velocity commands.

Control Strategy

- Distance control: PI controller to maintain a target following distance.
- Heading control: PD controller to keep the person centered in the field of view.
- Stability aids: Exponential smoothing, deadbands, anti-windup, slew-rate limits, and minimum-distance safeguards.

Project Planning

Week	Description	Objectives
1st Week	Project kickoff: define scope, risks, success metrics, milestones, and create the repository and backlog.	Approved scope and success criteria; repo + branching strategy set; prioritized backlog; dev environment ready (ROS 2 Humble + DepthAI).
2nd Week	Build the first perception prototype: train or select a YOLOv8 person model, export to ONNX, compile OpenVINO blob, and run it on the OAK-D to publish 3D person detections.	Real-time on-device detections (≥20 FPS) with XYZ published to ROS 2; basic diagnostics topic available.
3rd Week	Fine-tune and add target selection so the robot follows one specific person without confusing them with others (ID persistence, hand-off logic).	Stable single-target tracking with <25% ID swaps in simple multi-person scenes.
4th Week	Integrate the model on-camera and start basic robot motion: connect detections to velocity commands and verify safe movement end-to-end.	Robot approaches target and respects a safe minimum distance; watchdog + loss-of-target behaviors implemented.
5th Week	Design and tune the motion controller for smooth, controlled following (PI for distance, PD for heading, with limits and filters).	Smooth /cmd_vel with bounded accel/jerk; steady following at target distance without visible oscillations; gains configurable via YAML.
6th Week	Integrate all modules: neural network, tracker, and controller, with clean interfaces and launch files.	One-command launch for full stack; topics and params documented.
7th Week	Execute tests and final fine-tuning: corridors and moderate crowds; log edge cases and adjust thresholds/models.	Track maintained ≥75% of the time in moderate crowds; safe stop on target loss; functional test plan passed.
8th Week	Produce documentation, final report, short demo, analyze results and prepare final presentation.	Complete README + diagrams + test report; demo video recorded; backlog triaged with clear future work.