Workshop: Autonomous Clothing Pickup with TIAGo Pro

Participants work hands-on with a PAL Robotics TIAGo Pro mobile manipulator to build an autonomous pickup pipeline — the robot detects clothing on the floor using vision, navigates to it, picks it up, carries it to a drop-off point, and places it down.

The workshop is structured in two phases.

Phase 1: Exploring the subsystems

Participants start by running three standalone example scripts, each exercising one robot capability in isolation:

- **Vision** (`example_vision`) — Publishes a text prompt (e.g. "cloth on the floor") to a SAM3-based detection pipeline, receives a 6DoF grasp point in the camera frame, and transforms it to the map frame using TF2. Participants see how the robot perceives objects and learn coordinate frame transforms.
- **Navigation** (`example_navigate`) — Sends a goal pose to Nav2 and monitors progress via action feedback, cancelling the goal early when the robot is close enough. Participants learn the Nav2 action interface and can experiment with different goal positions on their map.
- **Arm control** (`example_arm`) — Runs a full pick-and-place cycle: raises the torso, looks down, publishes a target pose for the arm, calls the pick service, moves to a carry position, places, and returns home. Participants learn joint trajectory control, service calls, and the PlayMotion2 action for predefined poses.

Each example is self-contained and can be run independently against the real robot. An `EXAMPLES.md` guide in the workshop root covers all prerequisites (connecting to the robot, mapping, localization) and provides the exact commands to run each one.

Phase 2: Integration

Participants receive a skeleton `coordinator.py` — a ROS2 node with all boilerplate already in place (publishers, subscribers, service clients, state machine, approach pose calculation). Eight methods are left as TODO stubs that participants fill in by adapting code from the examples they just tested.

The TODOs are ordered by difficulty: simple joint commands first (near-copy-paste), navigation next, then arm service calls, and finally the TF2 transforms. An `INSTRUCTIONS.md` in the `task_coordinator` package provides the recommended order, a test command to trigger the pipeline without vision (via a manual topic publish), quaternion cheat-sheets, and a troubleshooting table.

What participants learn

- ROS2 fundamentals: topics, services, actions, TF2, QoS
- Integrating Nav2, MoveIt, and vision into one autonomous pipeline
- Coordinate frame transforms between camera, robot base, and map frames
- Incremental development and testing on real hardware
- Working with an existing codebase rather than starting from scratch

Infrastructure

- Docker container with all ROS2 dependencies pre-installed, mounting the source code from the host
- VSCode attached to the container for editing and terminal access
- Robot GUI available via browser for monitoring and manual control