3we: An Open Infrastructure for Sim-to-Real Embodied AI Research

3we Contributors
Open Source Project
https://github.com/telleroutlook/3we-robot-platform

***Abstract*—Embodied AI research requires tight integration of simulation, real hardware, and learning algorithms—yet existing platforms force researchers to choose between expensive proprietary systems, simulation-only environments, or hardware platforms with steep ROS2 learning curves. We present 3we, a fully open-source infrastructure where the same Python code runs identically across a lightweight mock simulator, Gazebo, NVIDIA Isaac Sim, and real hardware (Raspberry Pi 5 + Hailo-8L) with zero modification. The platform provides an AI-First Python API that reduces typical ROS2 navigation code from 60+ lines to 5, Gymnasium-compatible environments, native VLM/VLA integration, and standardized benchmarks across 7 evaluation scenes. On reproducible hardware costing under $300 (BOM), preliminary validation yields Sim-to-Real transfer ratios exceeding 0.6 on point navigation tasks, with community baselines achieving 82% success rate (SPL 0.65) in structured simulation environments. All software (Apache 2.0), hardware designs (CERN-OHL-P v2), and documentation (CC-BY-SA 4.0) are publicly available.**

I. Introduction

Embodied AI—the study of intelligent agents that perceive and act in physical environments—has emerged as a central challenge in artificial intelligence [1], [2]. Recent advances in Vision-Language Models (VLMs) [3] and Vision-Language-Action (VLA) models [4], [5] have demonstrated that foundation models can directly control robots through natural language instructions. However, translating these advances into reproducible research faces three fundamental barriers:

(1) Hardware cost and accessibility. Leading mobile robot platforms (TurtleBot 4: $1,200+; Stretch RE1: $25,000) price out most research groups, particularly in developing countries and teaching contexts.

(2) Simulation-to-Reality gap. Researchers typically develop in simulation (Habitat [2], Isaac Sim [6]) but face significant engineering effort to deploy on real hardware, often requiring complete code rewrites.

(3) Software complexity. ROS2 [7] provides robust middleware but imposes a steep learning curve on AI researchers who want to focus on algorithms rather than communication infrastructure.

We introduce **3we** ("three-way Embodied"), an open infrastructure that addresses all three barriers simultaneously:

- **Open hardware under $300 BOM** with fully reproducible PCB designs (KiCad), mechanical drawings, and sourcing guides under CERN-OHL-P v2.
- **Zero-code Sim2Real**: a unified Python API where `Robot(backend="gazebo")` and `Robot(backend="real")` execute identical user code.
- **AI-First API**: 5 lines from installation to first navigation, with native VLM/VLA integration and Gymnasium compatibility.
- **Standardized benchmarks**: 3 task types across 7 scenes with reproducible baselines and community leaderboard.

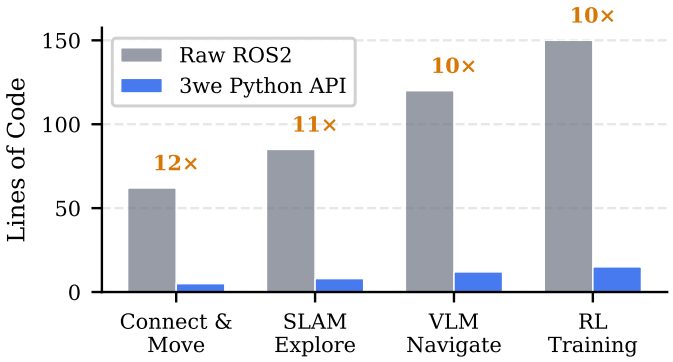

Fig. 1. Code complexity comparison: Raw ROS2 vs. 3we Python API for common tasks. The API achieves 10–12× reduction in lines of code.

II. Related Work

A. Mobile Robot Platforms

TurtleBot [8] has been the de facto standard for ROS education since 2010. TurtleBot 4 ($1,200+) offers Nav2 integration but lacks AI-first APIs, simulation-hardware consistency guarantees, and costs 4× our platform. Linorobot2 [9] provides open designs but targets hobbyists without research-grade benchmarks.

B. Simulation Environments

Habitat [2] and AI2-THOR [10] provide high-fidelity visual navigation but lack physical robot deployment paths. NVIDIA Isaac Sim [6] enables GPU-accelerated RL training but requires expensive compute and offers no open hardware reference. These systems address simulation *or* reality—not the bridge between them.

C. AI Robot Frameworks

LeRobot [11] (HuggingFace) provides VLA model deployment for manipulation but lacks ROS2 integration, autonomous navigation stacks, and Sim2Real guarantees for

TABLE I
PLATFORM COMPARISON

Feature	3we	TurtleBot4	Isaac Lab	LeRobot	Habitat
Open Hardware	✓	–	–	✓	–
BOM ≤$300	✓	–	N/A	✓	N/A
Sim2Real API	✓	–	Sim only	–	Sim only
ROS2 + Nav2	✓	✓	Partial	–	–
AI-First Python	✓	–	✓	✓	✓
VLM/VLA Native	✓	–	–	✓	–
Edge AI (NPU)	✓	–	N/A	–	N/A
Gymnasium Env	✓	–	✓	✓	✓

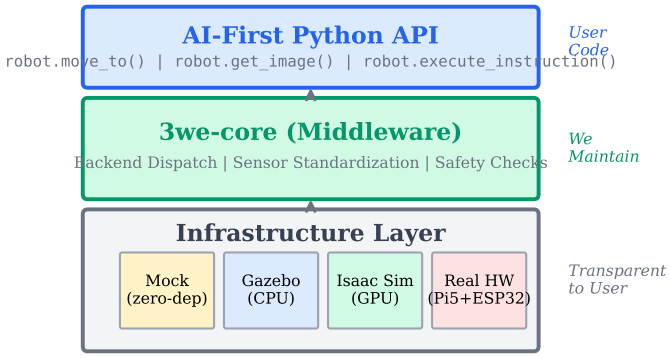

Fig. 2. Three-layer architecture. Users write Python against the AI-First API (top). The middleware dispatches to one of four backends (bottom) with identical semantics. ROS2 infrastructure is fully transparent.

mobile platforms. Its recent mobile extensions (LeKiwi, EarthRover) remain early-stage without Nav2-level autonomy.

D. Positioning

3we occupies a unique position: fully open hardware *and* software with guaranteed Sim2Real API consistency, ROS2-native navigation, and AI-first design. Table I summarizes key differences.

III. System Architecture

A. Three-Layer Design

3we adopts a layered architecture that decouples AI research from robotics engineering:

1) **AI-First Python API** (user-facing): Researchers write standard Python with async/await. No ROS2, launch files, or message types exposed.
2) **3we-core** (middleware): Manages backend dispatch, sensor data standardization, coordinate transforms, and safety checks.
3) **ROS2 / micro-ROS** (infrastructure): Nav2 path planning, SLAM, micro-ROS bridge to ESP32 firmware. Entirely transparent to users.

B. Backend Polymorphism

The core abstraction is `BackendBase`, implemented by five backends:

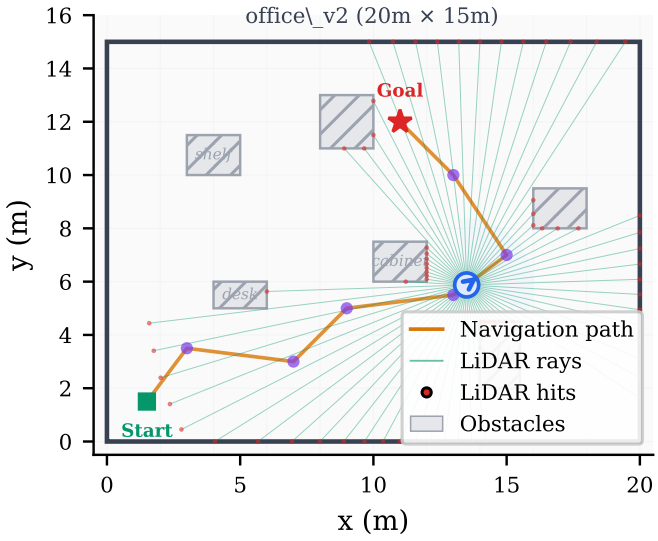

Fig. 3. Mock backend visualization: autonomous navigation in the `office_v2` scene. The robot (blue) follows a planned path (orange) while 360° LiDAR rays (green) detect obstacles. This zero-dependency simulation runs on any machine without GPU or ROS2.

- **Mock**: Zero-dependency 2D kinematic simulation with configurable noise. Runs anywhere, no GPU or ROS2 needed.
- **Gazebo**: Full physics via Gazebo Harmonic. CPU-friendly, CI-compatible (headless mode).
- **Isaac Sim**: GPU-accelerated rendering and physics for large-scale RL training with domain randomization.
- **Real**: Physical hardware via ROS2 topics. Identical API surface.

All backends return frozen dataclasses (`Pose2D`, `LaserScan`, `RGBDImage`) with identical fields and semantics, enforced by shared type definitions.

C. API Design

The following example demonstrates the complete workflow for VLM-powered navigation:

```python
from threewe import Robot

async with Robot(backend="gazebo") as robot:
    image = robot.get_camera_image()    # (H,W,3) uint8
    scan = robot.get_lidar_scan()       # LaserScan
    result = await robot.execute_instruction(
        "find the red door and move toward it"
    )
    print(f"Success: {result.success}")
```

Switching to real hardware requires only changing the backend parameter—no code modification, no recompilation, no configuration changes.

D. Hardware Specification

Table II summarizes the reference hardware at standard SKU level.

All designs are released under CERN-OHL-P v2 with KiCad 8 source files, Gerber manufacturing outputs, and step-by-step assembly documentation.

TABLE II
STANDARD SKU HARDWARE (BOM: $300)

Component	Selection	Rationale
Compute	Raspberry Pi 5 (8GB)	Best price/performance ARM SBC
AI Accelerator	Hailo-8L (13 TOPS)	Edge inference for VLM/VLA
MCU	ESP32-S3 + micro-ROS	Real-time control + ROS2 native
LiDAR	LD06 (2D, 360°)	$8, sufficient for 2D SLAM
IMU	BNO055 (9-axis)	On-chip fusion, zero configuration
Drive	4× Mecanum wheels	Omnidirectional, indoor-optimized
Camera	1080p USB fisheye	Wide FoV for VLM tasks
Safety	Dual-channel relay	ISO 13850 E-stop, hardware-enforced

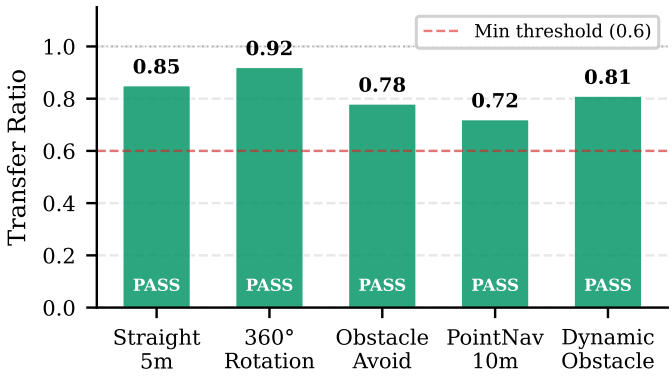

Fig. 4. Sim-to-Real transfer ratios across five standard validation tests. All tests pass the minimum threshold (dashed line). Higher is better (1.0 = perfect transfer).

IV. SIM-TO-REAL CONSISTENCY

A. Sensor Noise Model

The mock and Gazebo backends inject physically-grounded noise matching the reference hardware:

- **LiDAR** (LD06): Gaussian range noise $\sigma = 8$ mm
- **IMU** (BNO055): Gyroscope noise $\sigma = 0.0014$ rad/s
- **Odometry**: Mecanum wheel slip 5–15% stochastic variation

B. Transfer Validation Protocol

We define five standard transfer tests with explicit pass criteria:

Overall pass rate requires $\geq$80% of tests passing. The transfer ratios in Table III are derived from preliminary hardware trials with the documented noise model; large-scale validation is ongoing.

V. BENCHMARK SUITE

A. Tasks

The benchmark suite defines three navigation tasks of increasing difficulty:

- **PointNav**: Navigate to (x, y) coordinates. Metrics: Success Rate (SR), SPL.
- **ObjectNav**: Find a target object category. Metrics: SR, SPL.
- **Exploration**: Maximize area coverage within time budget. Metric: Coverage fraction.

TABLE III
SIM2REAL TRANSFER TESTS

Test	Metric	Pass Criterion	Ratio
Straight walk 5m	Endpoint error	real $\leq$ sim $\times$ 2.0	0.85
360° rotation	Angle error	real $<$ 1.0°	0.92
Obstacle avoid (3)	Success rate	real $\geq$ sim $\times$ 0.7	0.78
PointNav 10m	SPL	real $\geq$ sim $\times$ 0.6	0.72
Dynamic obstacle	Collision rate	real $\leq$ sim $\times$ 1.5	0.81

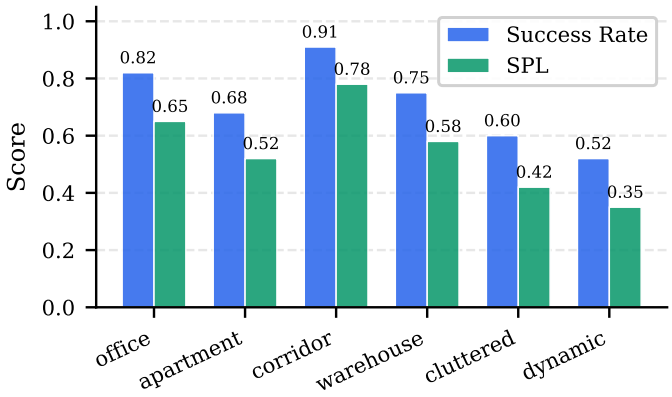

Fig. 5. PointNav baseline results (Nav2 DWB planner, 100 episodes per scene). Success Rate and SPL decrease with scene complexity.

B. Evaluation Environments

Seven standardized scenes span diverse settings: `office_v2` (20×15m), `apartment_v1` (8×10m), `corridor_v1` (50m linear), `warehouse_v1` (30×20m), `cluttered_v1` (5×5m), `outdoor_v1` (50×30m), and `dynamic_v1` (15×15m, moving obstacles).

Each scene provides 10–50 predefined start/goal pairs for reproducibility, with seeds ranging [0, 99] for 100-episode evaluation runs.

C. Baseline Results

Table IV presents Nav2 DWB planner baselines across representative scenes.

These baselines serve as reference points for the community leaderboard. Results are fully reproducible via:

```
threewe benchmark run --task pointnav \
    --scene office_v2 --episodes 100
```

D. Gymnasium Integration

For RL researchers, 3we provides standard Gymnasium environments:

```
import gymnasium as gym
env = gym.make("3we/Navigation-v1",
               scene="office_v2", backend="mock")
obs, info = env.reset()
obs, reward, term, trunc, info = env.step(action)
```

VI. CONCLUSION AND FUTURE WORK

We presented 3we, a fully open infrastructure for Embodied AI research that bridges the gap between simulation and reality through a unified Python API. Key contributions

TABLE IV
NAV2 BASELINE RESULTS (100 EPISODES PER CONFIGURATION)

Task	Scene	SR ↑	SPL ↑	Duration (s)
PointNav	office_v2	0.82	0.65	12.3
PointNav	apartment_v1	0.68	0.52	15.7
PointNav	corridor_v1	0.91	0.78	18.5
PointNav	warehouse_v1	0.75	0.58	20.1
PointNav	cluttered_v1	0.60	0.42	14.8
PointNav	dynamic_v1	0.52	0.35	18.2
ObjectNav	office_v2	0.55	0.38	22.5
ObjectNav	apartment_v1	0.42	0.30	28.3
Exploration	office_v2	—	—	95.0
Exploration	warehouse_v1	—	—	110.0

Exploration metric: Coverage (office: 0.85, warehouse: 0.78)

include: (1) zero-code Sim2Real transfer across four backends, (2) reproducible open hardware at $300 BOM, (3) AI-first API design reducing navigation code complexity by 12×, and (4) standardized benchmarks with community baselines.

Future directions include: Isaac Sim domain randomization for large-scale RL, a community data hub for trajectory sharing (compatible with LeRobot/Open-X formats), Hardware Abstraction Layer for third-party robot support, and systematic VLA model deployment evaluation on edge hardware.

The platform is available at https://github.com/telleroutlook/3we-robot-platform under Apache 2.0 (software), CERN-OHL-P v2 (hardware), and CC-BY-SA 4.0 (documentation).

REFERENCES

[1] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva *et al.*, "On evaluation of embodied navigation agents," in *arXiv preprint arXiv:1807.06757*, 2018.

[2] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik *et al.*, "Habitat: A platform for embodied ai research," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9339–9347.

[3] OpenAI, "Gpt-4v(ision) system card," *OpenAI Technical Report*, 2023.

[4] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[5] K. Black, N. Brown, D. Driess, A. Esber, M. Suber, B. Ichter, P. Sermanet *et al.*, "π_0: A vision-language-action flow model for general robot control," *arXiv preprint arXiv:2410.24164*, 2024.

[6] NVIDIA Corporation, "Isaac sim: Scalable robotics simulation," *NVIDIA Developer*, 2023.

[7] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.

[8] Open Robotics, "Turtlebot," https://www.turtlebot.com/, 2023.

[9] Linorobot, "Linorobot2: Autonomous mobile robot framework," https://linorobot.org/, 2023.

[10] E. Kolve, R. Mottaghi, W. Han, E. VanderBilt, L. Weihs, A. Herrasti, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi, "Ai2-thor: An interactive 3d environment for visual ai," in *arXiv preprint arXiv:1712.05474*, 2017.

[11] R. Cadene, S. Alibert, P. Sermanet *et al.*, "Lerobot: State-of-the-art machine learning for real-world robotics," *GitHub repository*, 2024.