

A Duckiebot as an Autonomous Environmental Sensor Scanner

Design, Validation and Failure Analysis of a Low-Cost Mobile Platform
for Geotagged Indoor Mapping

Michail G. Gkasios

Department of Informatics and Telematics
Harokopio University, Athens, Greece

August 2026

Workspace, source code and experimental data:
<https://gitlab.hua.gr/mobile-sensor-unit>

Abstract

We present the design, implementation and experimental evaluation of a Duckiebot DB21 modified into an autonomous environmental sensor scanner: a low-cost differential-drive robot that explores an indoor space unsupervised, builds a 2D occupancy map with a planar LiDAR, and simultaneously records temperature, humidity, pressure and air-quality readings geotagged to the map it is building. A scoping review of the Duckiebot and Duckietown literature is included in full (Section 2): it finds that although Duckiebot-derived platforms have been fitted with LiDAR and run conventional SLAM stacks, no peer-reviewed study evaluates a Duckiebot as an autonomous room scanner, and none couple such mapping to environmental sensing. The remainder of this report addresses the system-building and evaluation side of that gap.

Three contributions are offered. First, a complete open system: staged sensing bring-up, gmapping-based SLAM, a coverage-aware patrol planner that re-plans each pass over ground the sensors have not yet crossed, a hard time-of-flight safety gate independent of the planner, and a geotagging pipeline. Second, a validation methodology that scores the environmental reconstruction rather than displaying it: in simulation the sensor field is generated from a known $f(x, y)$, so the recovered map can be compared against the field that produced it, with two deliberately structureless fields as negative controls. Temperature recovers at $r = 0.996$ and eCO_2 at $r = 0.979$, while both controls remain near zero. Third, and most transferable, a failure analysis: we document and quantify a set of defects that a low-cost platform makes likely and that conventional map-quality metrics cannot detect. Chief among them, the LiDAR was mounted rotated 180° relative to its declared transform for the entire project. A controlled ablation in simulation shows this costs 4.5 m of localization error and reduces waypoint success from 100% to 3% in a corridor *while leaving mapped free area unchanged* — an area-based quality metric ranks the broken configuration marginally higher.

On hardware, correcting that transform took the system from 3/11 waypoints with 45 safety-gate blocks to four consecutive traverses with zero blocks and zero jams. A pre-registered experiment then tested whether the platform can detect a ventilation change: opening a door to an air-conditioned room is detected unambiguously in the *geometry* (the mapped wall recedes

1.7 m and returns to within 2 cm when closed), and in the *thermal balance* of the sensor (+0.200, −0.115, +0.077 °C/min across closed/open/closed). Two pre-registered predictions about a *localized* thermal anomaly failed, and are reported as failed; the spatial measurement was underpowered because the coverage planner optimises for reaching waypoints rather than for sampling space. We argue this last point is a design principle for mobile environmental sensing rather than a defect of this implementation.

Keywords: Duckiebot; mobile sensing; SLAM; occupancy mapping; coverage planning; indoor air quality; geotagged measurement; sensor extrinsics; failure analysis; low-cost robotics

Contents

1	Introduction	4
1.1	Motivating problem	4
1.2	The gap this addresses	4
1.3	Contributions	4
1.4	Scope and honesty statement	5
2	Related Work	5
2.1	Scope, questions and search method	5
2.2	Duckiebot hardware as a sensing platform	6
2.3	Evidence directly relevant to mapping and room scanning	7
2.4	Closely analogous indoor mapping evidence	8
2.5	Why a dedicated Duckiebot room-scanning literature is sparse	9
2.6	What else Duckiebots are used for	9
2.7	SLAM and navigation back ends used in this work	10
2.8	Synthesis: what is missing	10
3	System Description	11
3.1	Hardware	11
3.2	Software architecture	12
3.3	Safety gate	12
4	Method	12
4.1	Staged bring-up and exploration	12
4.2	Coverage-aware patrol planning	13
4.3	Environmental geotagging	13
4.4	Detrending the sensor’s self-heating	13
4.5	Evaluation metrics	13
5	Validation in Simulation	14
5.1	Coverage-aware re-planning	14
5.2	Field recovery against known ground truth	15
5.3	Ablation: a 180° sensor-extrinsics error	16
6	Hardware Results	16
6.1	Environment	16

6.2	Before and after the extrinsics correction	16
6.3	Environmental structure on hardware	19
7	Pre-Registered Experiment: Detecting a Ventilation Change	21
7.1	Design	21
7.2	Result 1: geometric detection of door state	21
7.3	Result 2: two failed predictions	21
7.4	Result 3: ventilation detected through the sensor’s thermal balance	22
7.5	Why the spatial measurement failed	22
8	Failure Analysis	23
8.1	The LiDAR was mounted backwards	23
8.2	Wheel odometry fabricates rotation	23
8.3	Sensors that fail silently	23
8.4	A calibration that could not be resolved	24
8.5	Retracted conclusions	24
8.6	Metric traps	24
9	Discussion	25
10	Limitations	25
11	Future Work	26
12	Conclusion	26
13	Code and Data Availability	27
A	Supplementary Tables	28
	References	31

1 Introduction

1.1 Motivating problem

Deciding how long to open the windows of an occupied classroom is a question with a real cost on both sides. Opened too briefly, accumulated carbon dioxide and volatile organic compounds are not cleared; opened too long in winter, the heat lost must be paid for again. The information needed to make that trade-off is not available from a single sensor on a wall, because the quantity of interest varies across the room: near the window, near the door, near the occupants, in the corners the air does not reach. What is wanted is a *spatial* description — how the measured quantity varies over the floor plan — and ideally how that description evolves while a window is open.

A mobile robot is an attractive instrument for this. One sensor carried through the space substitutes for many fixed sensors, it can be re-tasked to a different room, and the same platform that carries the sensor can build the floor plan the readings are attached to. The instrument is only useful, however, if it can be trusted: the readings must be attributed to the right places, and the system must not silently produce a confident description of a room it has mapped incorrectly.

This report is about building that instrument on deliberately cheap hardware and, more importantly, about finding out where it lies. The ventilation model itself is explicitly *not* attempted here and is stated as future work; what is delivered is the sensing platform and an honest account of its accuracy.

1.2 The gap this addresses

A scoping review of the Duckiebot and Duckietown literature [1, 2] examined whether the platform has been used to scan an indoor room. Its finding was that Duckiebots have a substantial record in lane following, perception, learning, multi-agent coordination, calibration and reproducible benchmarking, and that mapping appears in that record through AprilTag EKF-SLAM [3], centralized multi-camera SLAM [4], and project-level LiDAR builds running Hector SLAM and particle-filter SLAM [5, 6]. Peer-reviewed work confirms an RPLIDAR A1 integrated on a DB21, though used for driving perception rather than mapping [7, 8]. What the review did not find was a study whose central experiment is a Duckiebot autonomously scanning an ordinary indoor space and quantitatively evaluating the result.

Two further observations from that review shape this work. First, the scientific reason to choose a Duckiebot over a purpose-built mapping robot is that *the platform itself matters*: cost, standardisation, embedded compute limits and reproducibility. It follows that the interesting results from such a platform are as much about what breaks as about what works, and we treat the failure analysis accordingly — as a first-class contribution rather than an appendix. Second, the review distinguishes navigation-grade mapping from metrology-grade reconstruction. We adopt that distinction explicitly: nothing here should be read as a survey instrument.

1.3 Contributions

1. **A complete autonomous environmental scanning system** on a DB21 Duckiebot with an added planar LiDAR and an environmental sensor, comprising staged sensor bring-up, gmapping SLAM [9], frontier exploration [10] followed by a coverage-aware patrol, a planner-independent

time-of-flight safety gate, and a geotagging pipeline that attaches every reading to a pose in the map frame (Sections 3–4).

2. **Coverage-aware re-planning**, in which each patrol pass is planned over the free space the sensors have *not* already been carried across, together with a coverage metric $C(r)$ that measures what was sensed rather than what was navigated. Measured across seven simulated environments, this improves $C(0.25)$ by up to 39% relative to coverage-blind planning (Section 4.2).
3. **A scored validation methodology for the environmental map**. The simulated sensor is driven by a known spatial field, so the reconstruction can be compared against the field that generated it, with structureless fields included as negative controls (Section 5.2).
4. **A quantified failure analysis** of defects that low-cost platforms invite and standard metrics conceal, including a controlled ablation of a 180° sensor-extrinsics error (Sections 5.3 and 8).
5. **A pre-registered hardware experiment** on detecting a ventilation change, reported against predictions committed before the data was collected, including two that failed (Section 7).

1.4 Scope and honesty statement

Hardware validation was performed in a single corridor of 4.6×1.0 m containing five doors; no room of sufficient size was available. Room-scale behaviour is evaluated in simulation only, across seven environments. Every hardware number in this report comes from that one corridor and one robot, and should be read accordingly.

Where a result is confounded, underpowered or contradicted by a later measurement, this is stated in the text rather than resolved in favour of the system. Three claims made during development were subsequently retracted on evidence and are reported as such in Section 8, because the sequence of wrong conclusions is itself the most instructive part of the record.

2 Related Work

This section synthesises a scoping literature review of the Duckiebot and Duckietown ecosystem, conducted to answer a specific question: has a Duckiebot been used as a mobile sensor platform to scan an ordinary indoor room, and what does the surrounding literature establish about the platform’s suitability for that task? The review is folded into this report in full, rather than summarised and cited separately, because its conclusion — a documented gap between demonstrated feasibility and peer-reviewed evaluation — is the premise the rest of this report responds to.

2.1 Scope, questions and search method

The review addresses three questions: (RQ1) has a Duckiebot been used to scan or map an indoor room as a mobile sensor platform; (RQ2) what Duckiebot-based mapping and localization systems provide the closest technical precedents; and (RQ3) what other research tasks are Duckiebots used for in the scientific literature. It prioritises physical Duckiebot/Duckietown research over simulation-only studies, and includes non-Duckiebot SLAM papers only where they establish methods directly relevant to interpreting the room-scanning evidence.

It is a scoping review rather than a formal PRISMA systematic review, with a search cut-off of 16 August 2026, targeting combinations of the terms Duckiebot, Duckietown, SLAM, mapping, room/indoor mapping, LiDAR, Hector SLAM, occupancy grid, RTAB-Map, visual SLAM, AprilTag localization, point cloud, 3D mapping and sensor fusion across IEEE Xplore, ACM, ScienceDirect, Sensors/MDPI, arXiv, official Duckietown documentation, and university or author project pages. Because *scan a room* is not a standard bibliographic keyword, records were assessed by task semantics: a study counted as direct room-scanning evidence only if a Duckiebot or Duckiebot-derived platform acquired spatial measurements of arbitrary environmental geometry and produced a metric map, with tag-based localization classed as mapping-*adjacent* rather than equivalent. Evidence was graded into four tiers — peer-reviewed Duckiebot studies (A), official or primary-author project pages documenting physical builds but not peer-reviewed (B), university/student project records (C), and peer-reviewed non-Duckiebot SLAM work used as transferable precedent (D) — and conclusions about whether room scanning has *ever* been done draw on Tiers A–C, while claims about the strength of the scientific record emphasise Tier A only. The broad source mix improves sensitivity to “has it been done” at the cost of not every included item carrying equal scholarly weight; that trade-off is made explicit throughout by labelling each source’s tier rather than treating a project page as proof of a peer-reviewed finding.

2.2 Duckiebot hardware as a sensing platform

The Duckiebot was introduced as a deliberately minimal autonomous-vehicle platform: a small differential-drive robot with a single monocular camera and onboard compute, embedded in a structured road environment, that nonetheless demonstrated lane following, obstacle avoidance, localization in a global map, city-level navigation and multi-vehicle coordination on constrained hardware [1]. The platform’s importance was never sensor richness; it was running a full autonomy stack on cheap, standardised hardware. Later generations expanded the stack incrementally: DB19 added wheel encoders, and the DB21 family used here adds a time-of-flight sensor and IMU alongside camera and encoders, with Jetson Nano computation [2].

This matters for the room-scanning question because the stock sensor suite is not a panoramic scanning range sensor. A monocular camera can support visual SLAM, and encoders plus an IMU can stabilise pose estimation, but metric scale, texture dependence, illumination sensitivity and a narrow field of view make a stock-camera room scanner a harder engineering problem than a 2D LiDAR system would be. The literature nonetheless demonstrates that external sensing can be added without abandoning the platform: Li and Okhrin mounted an RPLIDAR A1M8 on a DB21 for real-world driving experiments [7, 8], and the KnightCar project explicitly extended a Duckiebot with 2D LiDAR for SLAM and navigation [5]. These precedents remove the strongest practical objection to a Duckiebot room scanner: the platform is physically and computationally extensible enough to host a range sensor and a ROS mapping pipeline, which is exactly the modification made for this report (Section 3). For full three-dimensional reconstruction of vertical geometry rather than a floor plan, a depth camera or 3D LiDAR would be needed; established RGB-D/visual SLAM back ends such as RTAB-Map and ORB-SLAM3 would be the appropriate targets for such an extension [26, 19], but no strong peer-reviewed example of that configuration on a Duckiebot was located.

2.3 Evidence directly relevant to mapping and room scanning

Table 10 in Appendix A ranks the following studies and projects by directness to the room-scanning question; the discussion below follows the same order.

Foundational map-based autonomy. Paull et al. established that a very low-cost platform could support substantial autonomy with only a monocular camera [1], demonstrating localization within a global map and city-level navigation on physical Duckiebots. It does not describe the Duckiebot as a general mapping robot discovering an unknown room: the environment is deliberately structured, with roads, lanes, signs and known visual conventions reducing the perception problem relative to arbitrary indoor geometry. The distinction between *using* a map and *building* one is crucial here — a robot that follows a lane in a known environment may localize accurately without ever generating an occupancy map of the surrounding walls.

EKF-SLAM on Duckiebots. An official University of Montreal project implements Extended Kalman Filter SLAM on DB21-J4 Duckiebots, jointly estimating robot pose and AprilTag landmark positions from wheel-encoder odometry and visual tag corrections, and explicitly frames the problem as SLAM under uncertainty, discussing real-hardware issues such as imperfect kinematics, camera distortion and latency [3]. It is a strong conceptual precedent — the Duckiebot maintains an evolving probabilistic map while estimating its own pose — but the representation is sparse and semantic: landmarks are discrete tags deliberately placed in the environment, whereas a room scanner must infer the geometry of previously uninstrumented walls, furniture and doors, which requires dense or semi-dense data and different data-association behaviour.

Centralized and multi-camera Duckietown mapping. The official Anatidaephilia project combines observations from Duckiebots and fixed “watchtower” cameras to estimate AprilTag and robot poses [4], and Magiera, Długosz and Skruch developed a peer-reviewed multi-camera visual localization system that stitches camera views through AprilTags into a top-down representation, reported as particularly useful for Duckietown [28]. Both broaden what “mapping” means in the Duckietown ecosystem — cooperative perception, global coordinate estimation, multi-robot localization — but are poor analogues of an onboard room scanner because sensing is partly or wholly external and the map boundaries are instrumented with fiducials; their relevance is architectural rather than direct.

KnightCar: the strongest documented 2D SLAM precedent. The KnightCar project is the most explicit example found of a Duckiebot-derived robot turned into a conventional mobile mapping platform: a low-cost ROS autonomous platform built by extending a Duckiebot with 2D LiDAR, running particle-filter SLAM, EKF-SLAM, Adaptive Monte Carlo Localization and A* path planning [5]. It substantially answers the feasibility question — the Duckiebot chassis, differential-drive kinematics and ROS ecosystem can support the same 2D range-based mapping pipeline used by general indoor mobile robots — but cannot be treated as equivalent to a peer-reviewed study: there is no formal room-scanning benchmark, ground-truth error analysis, or archival paper presenting the SLAM results as a research contribution. It is best classified as high-relevance technical evidence with medium scholarly strength.

Northeastern Hector-SLAM project. A Northeastern University project record documents a Duckiebot equipped with a monocular camera and SlamTech 2D LiDAR, implementing Hector SLAM in ROS to build a map and localize the robot, alongside standard lane-following and traffic-sign tasks [6]. This is notable because Hector SLAM is a mature fast-scan-matching algorithm family that can operate without wheel odometry [12] — the project is not merely adding a range sensor, it is running a genuine room-mapping algorithm. Its limitation is again evidentiary: the accessible source is a project profile, not a peer-reviewed paper with quantitative mapping results.

Peer-reviewed evidence of LiDAR integration on DB21. Li and Okhrin provide independent, peer-reviewed confirmation that a DB21 can be instrumented with a rotating 2D LiDAR: their real-world autonomous-driving framework mounted an RPLIDAR A1M8 on DB21 as an additional sensor [7, 8]. Their task was lane following, car following and overtaking, not mapping, so the output was not an occupancy map — but the paper reduces uncertainty about physical LiDAR integration on a contemporary DB21. Together with the project-level SLAM examples above, it indicates that the remaining gap is not whether the platform *can* carry a scanning LiDAR, but the lack of a systematic peer-reviewed evaluation of what it builds when it does.

2.4 Closely analogous indoor mapping evidence

The absence of a dedicated Duckiebot room-scanning paper is not evidence of technical uncertainty about the underlying robotics problem; indoor mapping with differential-drive robots is mature. The closest educational-robot analogue is EUROPA, a low-cost ROS-based differential-drive robot: Karalekas, Vologiannidis and Kalomiros describe a classroom lesson in which students used Hector SLAM to map their own classroom in RViz, discussing occupancy-grid construction by aligning laser-scan endpoints with the growing map [24] — essentially the experiment a LiDAR-equipped Duckiebot could reproduce, on a different educational platform.

Canonical mapping algorithms strengthen the transfer argument. Grisetti, Stachniss and Burgard’s Rao-Blackwellized particle-filter method, widely associated with gmapping, demonstrated grid mapping with real mobile robots at scale [9]; Kohlbrecher et al. presented Hector SLAM for rapid occupancy-grid mapping from LiDAR scan matching and inertial attitude estimation under limited compute [12]; Hess et al. presented real-time 2D LiDAR SLAM with loop closure, explicitly motivated as a method for acquiring as-built floor plans at 5 cm resolution on constrained hardware [22]. For visual and three-dimensional mapping, RTAB-Map provides a unified visual/LiDAR SLAM framework for long-term operation [26], and ORB-SLAM3 supports monocular, stereo, RGB-D and visual-inertial configurations validated indoors and outdoors [19]. These systems establish that the algorithms already exist; the open question is not *can* room mapping be done, but *how well can it be done on the Duckiebot platform*, given its sensor, compute, size, cost and reproducibility constraints. Autonomous scanning further requires exploration rather than mere mapping: Yamauchi’s frontier-based method drives a robot toward the boundary between known free space and unknown space until the environment is explored [10], which is the exploration strategy used in this report (Section 4) — no dedicated Duckiebot implementation using this approach was located in the reviewed literature.

2.5 Why a dedicated Duckiebot room-scanning literature is sparse

The literature gap is consistent with Duckietown’s design philosophy. Duckiebots were created to study autonomy in a small structured road environment, not to compete with general indoor service robots; the original platform deliberately minimised sensing to a single monocular camera [1], and most contributions concentrate on perception and control from camera images, sim-to-real learning, traffic interaction and reproducibility rather than geometric reconstruction. A second reason is that room mapping is comparatively well solved on platforms already designed around LiDAR or RGB-D sensors — a researcher whose primary contribution is a new SLAM algorithm has an easier experimental path on a TurtleBot-like base, and the scientific reason to choose Duckiebot specifically strengthens only when the platform itself is part of the point: low cost, standardisation, embedded resource limits, sim-to-real continuity, or comparison across identical robots. Third, the standard Duckiebot camera is optimised for the road scene; visual room scanning suffers from low-texture walls, illumination change, monocular scale ambiguity and narrow field of view in a way that a planar LiDAR, which directly measures wall geometry, does not. Finally, terminology obscures relevant work: Duckietown researchers write *localization*, *state estimation*, *navigation* or *SLAM* rather than *room scanning*, and project pages often demonstrate mapping without a formal paper. A scoping review therefore finds a stronger technical record than a title-only search would suggest, but a weaker peer-reviewed record than the broad term “SLAM on Duckiebot” might imply.

2.6 What else Duckiebots are used for

Table 11 in Appendix A summarises these application areas with representative sources; the discussion below follows the same order.

Vision-based autonomous driving. Lane following remains the platform’s canonical task, from the foundational classical perception-and-control demonstration [1] through Gupta and Easwaran’s computationally efficient lane-following algorithm, designed to leave scarce onboard resources available for additional safety functions [21]. Obstacle perception has been studied camera-only as well: Nubert et al. developed real-time visual obstacle detection using inverse perspective mapping on a physical Duckiebot and Raspberry Pi [29], reinforcing a recurring theme in this literature — methods selected or designed to fit limited onboard compute.

Perception beyond lane markings. Thiang and colleagues built a Duckietown-based robot that recognises traffic signs from a fisheye camera and executes the corresponding manoeuvre, quantifying recognition under different lighting and distractor conditions [32] — representative of Duckietown’s value as a controlled perception testbed where environmental variables can be manipulated systematically while retaining real camera, motion and lighting imperfections.

Reinforcement learning, imitation learning, and sim-to-real. Duckietown is a prominent small-scale embodiment platform for learning-based control: Almási, Moni and Gyires-Tóth trained a DQN-based agent evaluated in simulated and real Duckietown environments [18]; Kalapos et al. used PPO and domain randomization for vision-based lane tracking and collision avoidance, transferring a simulation-trained policy to a real small-scale robot [23]; Lőrincz, Szemenyei and Moni compared imitation-learning and sim-to-real techniques for camera-based lane following [27].

Multi-agent learning extends this: Candela et al. trained MAPPO policies in a multi-agent simulator modelled on the Duckietown testbed and transferred them to physical robots, reporting substantially improved reward over a rule-based method and a major reduction of the reality gap with domain randomization [20]; Sazanovich et al. trained imitation-learning policies on mixtures of real and simulated Duckietown data to improve generalisation [30].

Safety, calibration, and reproducibility. Yuhas et al. implemented an embedded out-of-distribution detector on a real Duckiebot integrated with an emergency-stop behaviour, reporting an 87.5% successful stopping rate on their braking test bed [34] — evidence that Duckiebots are used to investigate failure detection and safe fallback under embedded resource constraints, not only to maximise autonomy performance, which is the same motivation behind the safety gate in this report (Section 3). Krinkin et al. addressed autonomous calibration of Duckiebot cameras and wheels using a pre-marked area and autonomous motion to collect calibration data without continuous human intervention [25], relevant here because reconstruction accuracy depends on calibrated wheel motion, sensor extrinsics and time synchronisation — exactly the class of defect examined in Section 8. Tani et al. developed DUCKIENet and the Duckietown Autolab for reproducible evaluation across simulation, logs and remote physical hardware, reporting low variance across robot hardware and remote labs [31], a standardisation emphasis relevant when comparing mapping results across implementations.

2.7 SLAM and navigation back ends used in this work

The algorithms this report builds on are themselves mature and are cited here rather than in the sections above because they are general robotics literature, not Duckiebot-specific. Occupancy grids date to Elfes [11]; Rao-Blackwellized particle-filter grid mapping (gmapping) provides odometry-conditioned 2D SLAM [9]; Hector SLAM offers scan-matching without an odometry prior [12]. Navigation rests on the dynamic window approach [13] and the ROS navigation stack [14, 15], and exploration on frontier selection [10]. Coverage path planning has its own literature [16]. The contribution of this report is therefore not a new back end but the integration of existing back ends, their coupling to environmental sensing, and an honest evaluation of where that integration breaks.

2.8 Synthesis: what is missing

Bringing the evidence together: a Duckiebot has been used for mapping and SLAM, and Duckiebot-derived robots have been equipped with scanning LiDAR for 2D SLAM (Section 2.3), but a dedicated, peer-reviewed study of a physical Duckiebot scanning a normal indoor room into a complete 2D or 3D model was not located in the searched literature. For the specific claim “a Duckiebot has been used as a sensor bot to scan a room,” the strongest defensible answer is *yes at the project/implementation level, but not established as a peer-reviewed research topic*: KnightCar and the Northeastern Hector-SLAM project are directly on point [5, 6], peer-reviewed LiDAR integration on DB21 makes those reports more plausible and transferable [7, 8], and official Duckietown SLAM projects show that mapping itself is already within the platform’s research scope [3, 4]. What is missing is a publication that brings these threads together and evaluates room reconstruction rigorously, coupled to a task the room geometry is actually useful for — which is the gap this report addresses (Section 1.2). A further methodological caution carries forward into this report’s own claims: “room scan” can

mean either navigation-quality mapping or metrology-quality reconstruction, and a low-cost 2D LiDAR Duckiebot should be judged against the former. Useful robotic maps are not equivalent to professional laser scans, and nothing in this report should be read as a survey instrument (Section 10).

3 System Description

3.1 Hardware

The base platform is a Duckietown DB21 with NVIDIA Jetson Nano computation. Stock sensing used here comprises wheel encoders, a front-facing time-of-flight (ToF) range sensor and an IMU. Two additions were made:

- **RPLIDAR C1M1-R2** planar LiDAR over USB, at 460 800 baud (*not* the 115 200 of older A-series units), publishing 860 beams per revolution at approximately 10 Hz with a usable range to several metres.
- **DFRobot SEN0335** environmental sensor (ENS160 + BME280) on the Jetson’s I²C bus, providing temperature, relative humidity, barometric pressure, an air-quality index, total volatile organic compounds (TVOC) and equivalent CO₂ (eCO₂) at 1 Hz.

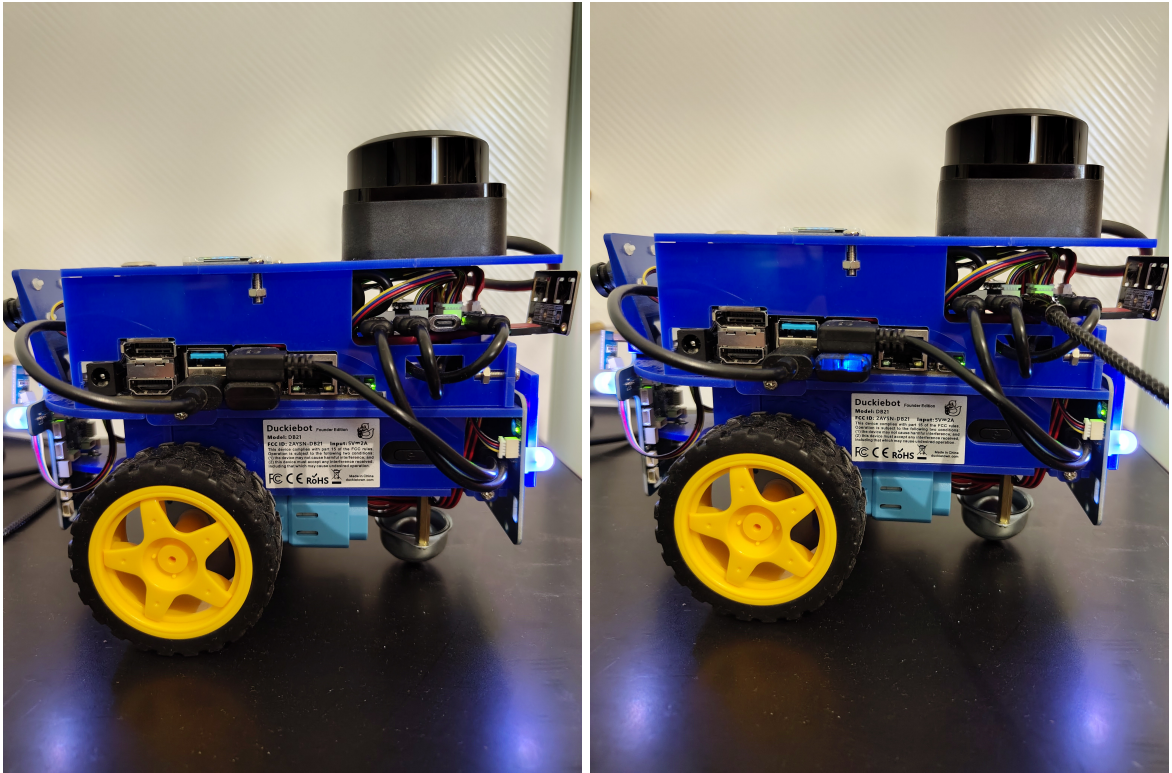


Figure 1: The DB21 with the added RPLIDAR C1M1-R2 (top) and DFRobot SEN0335 environmental sensor, rear view. Both additions sit above the stock chassis and are purely additive to the standard sensor suite; right: the same unit on its charging dock between runs.

The LiDAR is mounted 0.085 m *behind* the wheel axle and 0.16 m above the floor, both measured rather than assumed. The significance of the third extrinsic parameter — its yaw — is the subject of Section 5.3.

3.2 Software architecture

The system runs ROS 1 Noetic [14] in Docker containers alongside the stock Duckietown stack, sharing a single ROS master. It is purely additive: no stock node is replaced. The pipeline is

LiDAR $\rightarrow$ `/scan` $\rightarrow$ gmapping $\rightarrow$ `/map` and `map` $\rightarrow$ `odom` $\rightarrow$ `move_base` $\rightarrow$ `cmd_vel` $\rightarrow$ ToF gate $\rightarrow$ wheels,

with wheel odometry supplying the `odom` $\rightarrow$ `base_footprint` transform that gmapping consumes as its motion prior, and a measurement logger writing one geotagged CSV row per sensor sample by looking up `map` $\rightarrow$ `base_footprint` at the time of the reading.

3.3 Safety gate

Forward motion is gated independently of the planner. A node between `move_base` and the motor command topic refuses any positive linear velocity while the front ToF reads below 0.15 m, regardless of what the planner computed. Reverse and rotation are not gated, so the robot can always escape. This is deliberately not implemented as a costmap layer: a costmap informs planning, whereas this must hold when planning is wrong, which is precisely the case that matters.

The gate’s value was demonstrated when it fired 105 times in one early run, and its failure mode was demonstrated on a later date when a stale ROS registration left the ToF topic silent while the container still reported healthy — the gate failed *open* and the robot drove into a wall with zero blocks logged. Both are discussed in Section 8; the present system verifies ToF liveness before any autonomous run.

4 Method

4.1 Staged bring-up and exploration

Early hardware runs failed because navigation began before sensing was stable. The launch is therefore staged: sensors first, verified healthy, then SLAM, then navigation. Exploration itself is delayed 45 s after `move_base` starts so the map settles before anything commands motion. This ordering is not cosmetic. Measured in simulation on the `classroom` world, with only the ordering changed, it took the run from 2169 planner failures and 0.456 m of position error to 121 failures and 0.082 m.

Frontier exploration [10] runs first to reveal the space. It is, however, the wrong objective for a scanner: it terminates as soon as the map has no frontiers left, which in an open room can occur having driven the robot almost nowhere, leaving the environmental sensor having sampled a fraction of the floor. Exploration is therefore followed by a coverage patrol, and the handover is explicit — the patrol waits for the explorer to exist, then terminates it and verifies it is gone, because an earlier silent handover race left both fighting for the same action server and produced 9 of 16 goals preempted.

4.2 Coverage-aware patrol planning

The patrol plans a boustrophedon sweep over known-free space eroded by robot radius plus a safety margin. Where that erosion consumes the whole space — as it does in a corridor, which cannot provide clearance on both sides of a row — the planner falls back to a centreline traverse computed by an octagonal distance transform and a graph-diameter double breadth-first search, resampled onto the local clearance maximum. The fallback is chosen by comparing waypoint counts, not by a hard-coded environment type.

Patrol runs in multiple passes. The contribution here is that each pass is planned over the free space the sensors have *not* already been carried across: a covered mask is accumulated from the recorded trajectory by marking discs of radius r_{cov} around every recorded pose, and the next sweep is built on $\text{safe} \wedge \neg \text{covered}$. Passes stop when a pass grows the known map by less than 5%.

Two implementation faults are worth recording because both were silent. The map used for re-planning was initially served from a cache, so every pass re-planned from an identical map and reported exactly +0.0% growth. And waypoints were keyed by index, which restarts each pass, so later passes overwrote earlier ones in the rendered figures — 50 waypoints drawn against 90 counted.

4.3 Environmental geotagging

Each sensor sample is written with the pose obtained by looking up `map→base_footprint` at that moment, producing rows of `(t, x, y, yaw, temp, humidity, pressure, aqi, tvoc, eco2)`. Rendering interpolates these onto the occupancy grid by inverse-distance weighting [17], with one deliberate restriction: cells further than 0.35 m from any sample are left uncoloured. Interpolating a point measurement across the whole room manufactures a smooth, confident picture of places the sensor was never carried past. The coloured area is therefore the coverage claim, and it is meant to look as ragged as the run really was.

4.4 Detrending the sensor’s self-heating

The temperature channel reads the sensor’s own board heat as well as the air. On hardware it correlates $r = 0.92$ with elapsed time and only 0.35 with position: plotted raw it maps the robot warming up, with a hot region wherever the run happened to end. Analysis therefore removes the linear-in-time component before mapping.

That is only legitimate when position and time are not themselves confounded. If the robot sweeps the space once, monotonically, the two are inseparable and detrending would remove real spatial signal. The tool computes $r(x, t)$ and refuses to detrend when it is high. On the corridor runs the robot traverses back and forth, giving $r(x, t) = 0.12$, and detrending then *raised* the spatial correlation of temperature from 0.35 to 0.61 — the drift was masking structure, not substituting for it.

4.5 Evaluation metrics

Coverage $C(r)$. The fraction of free cells in the final map within r of some robot pose. Reported as a curve, because the defensible value depends on how far a point measurement may honestly be interpolated, which is a property of the sensed quantity rather than of the robot. $C(0.25)$ — half the

sweep row spacing, the finest scale the plan attempts — is used as the headline. Waypoints reached is reported separately and is *not* treated as coverage: a run can reach every waypoint of a bad plan.

Map quality. Walled extent (the bounding box of occupied cells) and occupied-cell count, checked against measured ground truth, and judged by eye. Free area is reported but explicitly *not* used as a quality score, for reasons Section 5.3 makes concrete.

Localization error. In simulation, against Gazebo’s true model pose. On hardware no such reference exists, which is the central epistemological problem of this work: every hardware localization figure is the pose estimate describing its own quality.

Field recovery. In simulation the environmental field is generated from a known $f(x, y)$, and reconstruction is scored by mean absolute error and by the correlation between logged values and true field over the visited points.

5 Validation in Simulation

Simulation uses Gazebo with the same ROS nodes as hardware; only the sensor drivers and a command bridge are substituted. Seven environments are used: a plain room, a classroom with a desk grid, a living room with sparse furniture, two rooms joined by a narrow doorway, two idealised corridors, and `corridor_real`, whose wall profile is derived directly from a hardware occupancy grid so that its doorways and width variation match the physical corridor.

5.1 Coverage-aware re-planning

World	Passes	Reached	$C(0.25)$ aware	$C(0.25)$ blind
<code>corridor_real</code>	2	22/22	0.543	0.550
<code>corridor_wide</code>	1	11/11	0.346	0.324
<code>corridor_railing</code>	1	11/11	0.352	0.362
<code>room</code>	1	16/16	0.653	0.545
<code>classroom</code>	3	75/76	0.588	0.456
<code>living_room</code>	2	60/61	0.726	0.712
<code>two_room</code>	3	—	0.561	0.403

Table 1: Coverage-aware versus coverage-blind patrol planning.
Corridors run a single pass under either policy, so the exclusion is inert there, as expected.

Room-like environments improve substantially — `two_room` by 39%, `classroom` by 29%, `room` by 20% — while corridors are unchanged because they never run a second pass. The full coverage curves show how strongly the headline number depends on the chosen radius:

World	Free m ²	Path m	$C(0.10)$	$C(0.25)$	$C(0.50)$	p_{95} m	sweep eff.
corridor_real	4.22	9.06	0.233	0.543	0.937	0.52	0.48
corridor_wide	7.91	5.71	0.143	0.346	0.596	1.12	0.90
corridor_railing	8.04	5.71	0.141	0.352	0.596	1.15	0.93
room	18.71	35.29	0.329	0.653	0.898	0.60	0.69
classroom	50.46	116.83	0.337	0.588	0.769	1.29	0.51
living_room	42.24	94.75	0.375	0.726	0.905	0.61	0.65

Table 2: Coverage metrics. p_{95} is the 95th-percentile distance from a free cell to the nearest pose; sweep efficiency is the share of the swept corridor that was new ground.

At $r = 0.10$ m no environment exceeds 0.38: a single planar sensor carried along a sparse path does not densely sample a room, and reporting only a generous radius would obscure that.

5.2 Field recovery against known ground truth

The simulated environmental sensor originally produced a random walk in *time* with no dependence on position. A heatmap built from such a sensor shows structure, and that structure is entirely an artefact of which places the robot visited at which moment: it can exercise the plumbing but can never demonstrate that a spatial signal is recovered, because there is no spatial signal. It was replaced with a known field — a linear gradient plus a Gaussian hot spot for temperature, a gradient along y for eCO₂ — sampled at Gazebo’s ground-truth position, with humidity and TVOC left as pure temporal noise to serve as negative controls.

World	Field	MAE	RMSE	Recovery r
corridor_real	temperature	0.094	0.116	0.999
corridor_real	eCO ₂	12.5	14.3	0.226
living_room	temperature	0.082	0.107	0.996
living_room	eCO ₂	17.6	19.3	0.979
<i>Negative controls (no spatial field; must not correlate)</i>				
corridor_real	humidity / TVOC	—	—	−0.302 / +0.014
living_room	humidity / TVOC	—	—	+0.001 / +0.167

Table 3: Reconstruction scored against the field that generated it.

Temperature recovers essentially perfectly in both environments. The poor eCO₂ score in the corridor is a measurement-design artefact rather than a pipeline failure: that field varies along y , and the corridor spans only 0.2 m of y , giving 11 ppm of true variation against ± 25 ppm of injected noise. Given a room with 5 m of y , the same field recovers at $r = 0.979$.

The controls carry more weight than the positive results. A pipeline that recovered structure from a

field that has none would do the same on hardware, where nothing could contradict it. Failing to recover a signal that is not there is the stronger evidence.

5.3 Ablation: a 180° sensor-extrinsics error

Section 8 describes the discovery that the physical LiDAR was mounted rotated 180° relative to the transform that placed its returns. The hardware before/after was dramatic but not controlled: battery charge, the generated plan and a second software change all differed between those runs. Simulation isolates the variable. The URDF sensor stays at 0° while the transform is told it is π — exactly the hardware fault — injected into worlds where both true geometry and true robot pose are known.

World	LiDAR	Reached	Position error	Free m ²	Occupied cells
corridor_real	correct	100%	0.021 m	4.18	752
corridor_real	reversed	3%	4.463 m	4.27	563
living_room	correct	99%	0.126 m	42.17	3220
living_room	reversed	0%	6.806 m	19.87	1362

Table 4: Controlled ablation of the sensor yaw error, two repeats per arm.

Navigation collapses in both environments. The instructive difference is in the *map*. In the living room the map itself falls apart — free area halves — so the fault is visible on sight. In the corridor the map remains clean, correctly shaped and correctly sized while the robot is 4.5 m from where it believes it is. Mapped free area is in fact marginally *larger* in the broken arm (4.27 against 4.18 m²): a quality metric based on mapped area would rank the broken configuration higher.

This yields a methodological claim beyond the specific bug: **map-quality statistics cannot detect a sensor-extrinsics fault; navigation success and ground-truth localization can.** It also explains why the fault survived weeks of debugging — the environment available for testing was the one that conceals it. A complementary offline test quantifies the concealment directly: scoring a reversed sensor’s scan *without* the rotation still matches the map at 0.939 against 1.000 for the correct orientation, a difference no threshold would flag.

6 Hardware Results

6.1 Environment

All hardware runs took place in a corridor measuring approximately 4.6×1.0 m containing five doors. The doors matter: they are the only along-axis features available to the scan matcher. A perfectly smooth corridor is degenerate for localization along its length — an offline test on a synthetic smooth corridor shows a 0.30 m longitudinal pose error still matching the map at 1.00, while the same error laterally scores 0.03. The recesses break that degeneracy, and the mapped wall profile recovers them.

6.2 Before and after the extrinsics correction

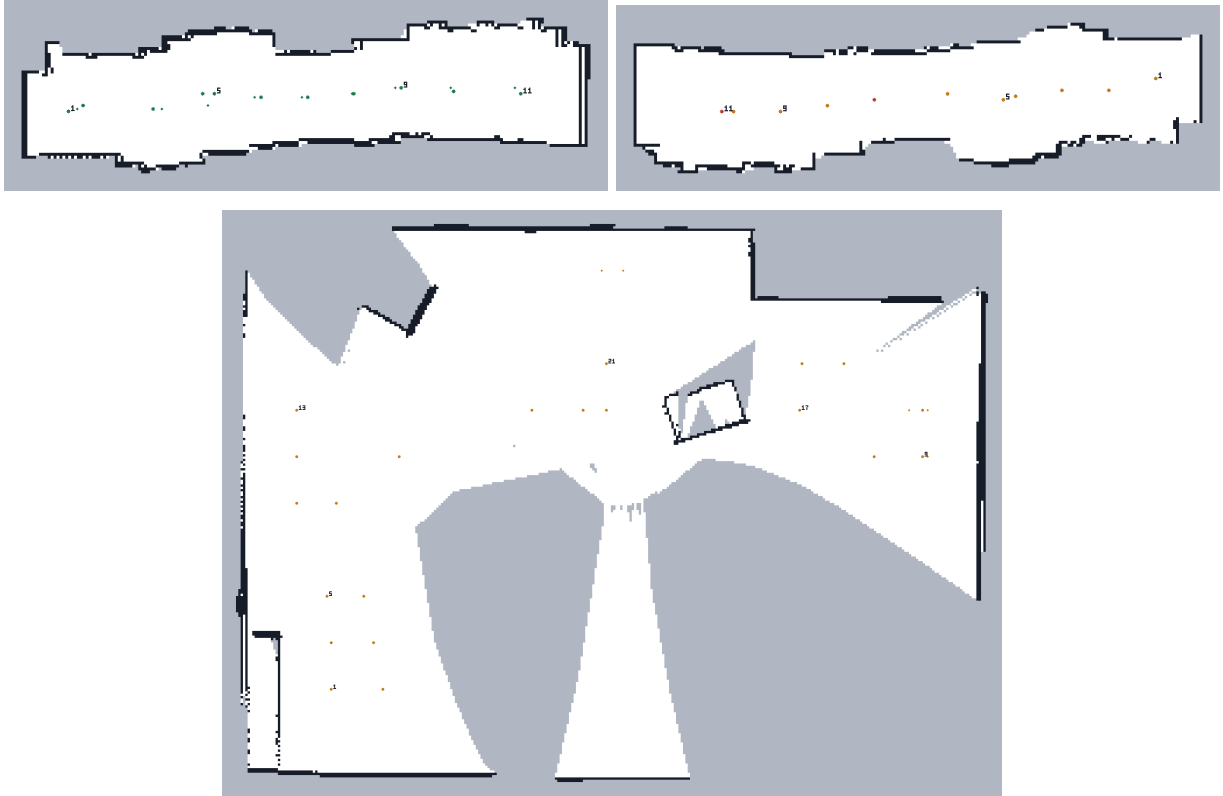


Figure 2: Controlled ablation of the 180° sensor-yaw error. Top left: corridor, correct transform, 21/21 waypoints, 0.021 m localization error. Top right: the same corridor with the transform reversed — a clean, correctly shaped, correctly sized map produced while the robot is 4.5 m from where it believes it is, reaching 0 of 11 waypoints. Bottom: the living room with the same fault, where the map itself collapses and the failure is visible on sight. The corridor does not tolerate the fault; it conceals it.



Figure 3: The corridor used for every hardware run in this report, looking along its length. The doors visible here are the along-axis features referred to throughout this section.

Run	Waypoints	Gate blocks	Jams	Free m ²	Map
2026-08-16 (pre-fix)	3/11	45	1	7.87	corridor drawn twice
2026-08-17 (post-fix)	17/19	0	0	5.02	single clean corridor
2026-08-18 run 1	22/22	0	0	—	clean
2026-08-18 run 2	11/11	0	0	—	clean
2026-08-18 run 3	11/11	0	0	—	clean

Table 5: Hardware traverses before and after correcting the LiDAR yaw. Free area falls because the pre-fix map drew the same space twice; true floor area is approximately 4.6 m².

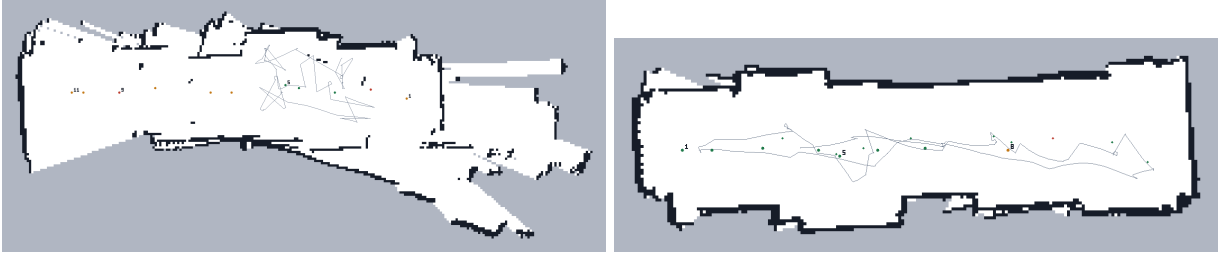


Figure 4: Hardware corridor maps with planned waypoints and the robot’s real trajectory. Left: before the extrinsics correction — the corridor is drawn twice, offset, with a phantom arm; 3 of 11 waypoints reached and 45 safety-gate blocks. Right: after correction — a single clean corridor, 17/19 waypoints, zero gate blocks, zero jams. Paths are drawn from the recorded trajectory, never as straight lines between waypoints.

Four consecutive traverses with zero safety-gate blocks and zero jams followed the correction, against a prior record in which the robot regularly drove into walls. The reduction in mapped free area from 7.87 to 5.02 m² is an improvement: the earlier figure was inflated by mapping the same corridor twice, and 5.02 against a true ≈ 4.6 m² is consistent with wall thickness plus a doorway where the corridor genuinely widens.

6.3 Environmental structure on hardware

On the 2026-08-17 run (464 geotagged samples over 3.84 m), after detrending as described in Section 4.4:

Field	$r(\text{time})$	$r(x)$ raw	$r(x)$ detrended
temperature	0.92	0.35	0.61
humidity	−0.31	−0.36	−0.35
TVOC	0.19	0.12	0.10
eCO ₂	0.20	0.10	0.08

Table 6: Spatial versus temporal structure in the hardware environmental data. $r(x, \text{time}) = 0.12$ for this run, so the two are separable.

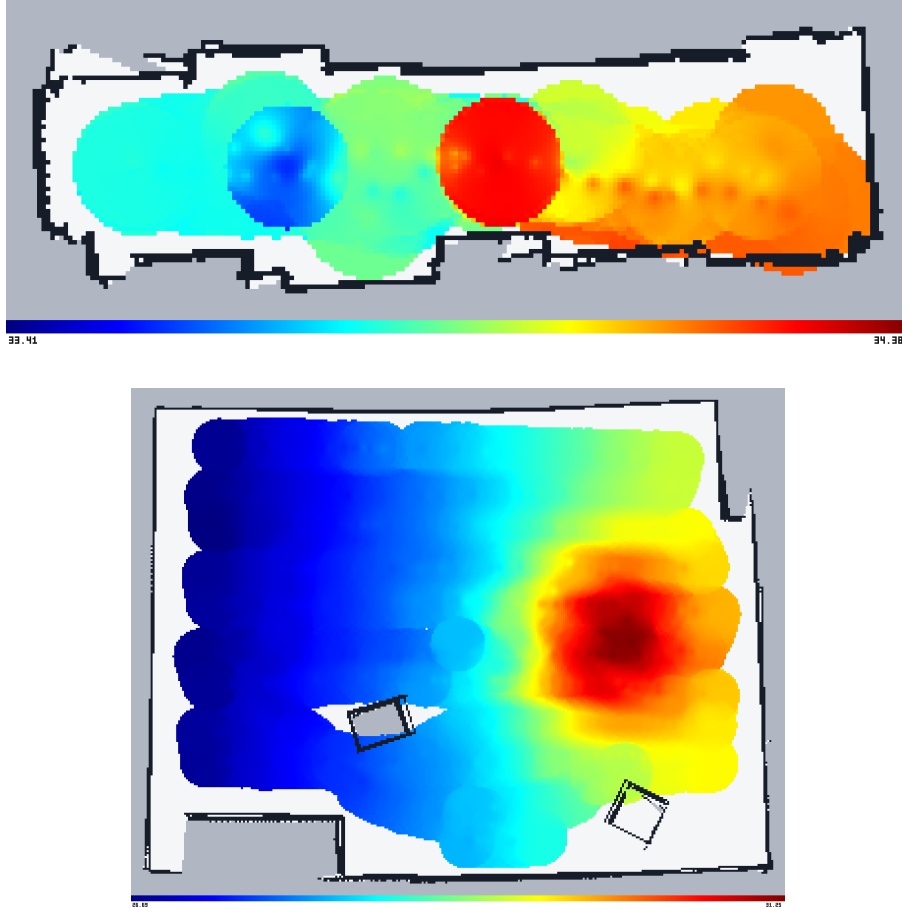


Figure 5: Environmental maps drawn on the occupancy grid the robot built. Top: hardware corridor temperature, detrended against time; the cool region at left lies at the door of an air-conditioned room. Bottom: simulated living room, where the field is known exactly and is recovered at $r = 0.996$; the Gaussian hot spot is reconstructed at its true location. Cells further than 0.35 m from any reading are left uncoloured, so the coloured area is the coverage claim.

Temperature and humidity carry spatial signal; TVOC and eCO₂ do not. The latter is reported as a genuine negative: a short corridor is well mixed for gaseous species, and claiming an air-quality map from this data would not be supportable. Only the temperature and humidity channels are claimed.

7 Pre-Registered Experiment: Detecting a Ventilation Change

7.1 Design

One of the five corridor doors opens onto an air-conditioned room. The 2026-08-17 run had independently produced two coinciding observations: a recess in the mapped south wall at $x \approx 0$, and the run’s coldest detrended temperature at the same location. Because that coincidence was uncontrolled — and because the robot’s start position, where it dwells longest, is also at $x \approx 0$ — a controlled experiment was designed.

Predictions, falsification criteria and the analysis method were committed to version control *before* any data was collected. Three runs were performed: door closed, door open with the air conditioning running, and door closed again, the last controlling for time-of-day drift.

7.2 Result 1: geometric detection of door state

Run	Door	Mapped south wall at the door
1	closed	−0.64 m
2	half open	− 2.32 m
3	closed	−0.66 m

Table 7: Door-state detection from LiDAR geometry alone.

The mapped wall recedes 1.7 m when the door is open — the sensor seeing through the aperture into the room — and returns to within 2 cm of its closed value when the door is closed again. Nothing in the system was told a door existed. The aperture was in fact only *half* open during the run (the operator reduced it so the robot would not drive into the room), making the detection more demanding than designed.

7.3 Result 2: two failed predictions

The pre-registered predictions concerned a *localized* thermal anomaly at the door. Both failed.

With the door closed, the anomaly at $x \approx 0$ was +0.45 °C in the first half of the run, decaying to +0.17 °C in the second — present and decaying, not weak as predicted. The decay supports an alternative mechanism identified before the run: residual cold air admitted when the operator opened the door to position the robot, pooling at the robot’s start position.

With the door open, $x \approx 0$ read 0.21 °C *warmer* than the corridor’s far end — the opposite of the prediction. The operator subsequently clarified that the door had been fully open during the five-minute settling period and only half open during the run, so the corridor was already thermally

mixed before measurement began. That is consistent with the observation, but it was known only afterwards and is therefore an explanation rather than a prediction.

The spatial measurement was also underpowered. Restricting to samples taken while the robot was actually moving leaves only 49–96 of roughly 420 samples per run, with some 0.25 m bins containing a single reading, against effect sizes of ± 0.2 °C.

7.4 Result 3: ventilation detected through the sensor’s thermal balance

An unanticipated result emerged from the quantity the analysis had been treating purely as a nuisance. The sensor’s self-heating rate is a probe of ambient cooling: a fixed heat source in cooler airflow warms more slowly.

Run	Door	Self-heating slope (°C/min)
1	closed	$+0.200 \pm 0.007$
2	half open	-0.115 ± 0.015
3	closed	$+0.077 \pm 0.002$

Table 8: Fitted temperature slope over each run. Door state moves it by 0.315 °C/min, roughly ten times the signal-to-noise of the spatial measurement.

The sign reverses when the door is closed again, which is what the third run was for: it rules out monotonic time-of-day drift as the cause. The recovery is incomplete in magnitude ($+0.077$ against $+0.200$), most plausibly because the corridor’s thermal mass remained cold; a single cycle cannot separate residual cooling from drift, and this is left unresolved.

An important limitation of this result. The measurement is temporal and global. It does not require the robot to move, and a fixed sensor would have detected it equally well. It is therefore evidence that the platform senses ventilation state — which bears directly on the motivating application — but it is **not** evidence for the mobile-mapping capability. That claim rests on the geometric results, not on this one.

7.5 Why the spatial measurement failed

In run 2 the robot drove for 47 s of a 414 s run; 298 of 415 samples came from a single stationary position. The coverage patrol is built for navigation — reach a waypoint, pause, re-plan — and that behaviour produces sparse, dwell-biased spatial sampling. Wherever the robot pauses dominates the statistics.

We take this to be the principal reason the spatial measurement failed, and it generalises: **a robot intended as a mobile environmental sensor should traverse continuously at constant speed, because a planner optimised for reaching waypoints is not optimised for sampling space.** These are different objectives that happen to share a vehicle.

8 Failure Analysis

This section is presented as a primary contribution. The defects below were found on a platform chosen partly because its constraints are representative, and several were invisible to the metrics conventionally used to judge robotic mapping.

8.1 The LiDAR was mounted backwards

The static transform placing LiDAR returns declared a yaw of 0. The sensor was physically mounted rotated 180°, so every scan was placed rotated half a turn about the sensor — obstacles ahead mapped behind — while wheel odometry reported true forward motion. SLAM was being handed two mutually inconsistent descriptions of the world.

It was found by accident. The operator reported more than 2 m of clear floor ahead; the LiDAR reported 1.28 m. Driving the robot 0.44 m forward made the range at bearing 0 *grow* and the range at 180° *shrink*. Two hypotheses fit that observation equally well — a reversed sensor, or a robot that drives backwards on a positive command — and they require opposite corrections, so the question was put to the operator rather than inferred.

The mount’s x , y and z had all been measured. Only the yaw had been assumed. Section 5.3 quantifies the cost and, more usefully, shows that a corridor conceals it.

8.2 Wheel odometry fabricates rotation

Encoder ticks measure the *wheels*’ rotation, which equals the *robot*’s only while the wheels grip. Two calibration runs with the operator observing:

Run	Wheel odom	Gyro	SLAM	Observed
turn 1	214.5°	43.9°	44.8°	“rotated slightly to the left”
turn 2	30.5°	−0.4°	2.5°	“didn’t move at all”

Table 9: In-place rotation as reported by three sources against direct observation.

The second row is the clearest statement of the problem: 30.5° of reported rotation for a robot that did not move. No multiplicative calibration constant repairs this, since no constant times zero yields 30.5. Heading was therefore taken from the gyroscope and only distance from the encoders. The split follows the geometry: differential drive derives heading from the *difference* of the two wheels, which is exactly what slip destroys, while the *sum* survives in-place slip largely intact.

This mattered beyond odometry, because gmapping consumes this signal as its motion prior.

8.3 Sensors that fail silently

Two independent instances, both of the same dangerous class — a sensor that stops reporting reality while continuing to report *something*:

- The front ToF topic fell silent because of a stale ROS registration while its container still reported healthy. The safety gate failed *open* and the robot drove into a wall with zero blocks logged.
- The IMU driver published 24 Hz of exactly 0.000000 while its container reported healthy and the chip still answered on the I²C bus; only its log admitted a communication loss.

Both now have explicit liveness checks. The IMU case is instructive because exact zeros are distinguishable from a stationary robot: a live gyroscope at rest reports noise, never a run of identical zeros. Detecting that requires knowing what a healthy sensor’s *noise* looks like, not merely whether messages arrive.

8.4 A calibration that could not be resolved

The robot veers measurably while driving straight. Using gyro-measured heading change per metre, six legs at trim = 0.13 gave $+31.9$ deg/m (sd 30.2) and six at 0.10 gave $+21.6$ (sd 35.0). The difference, $+10.3 \pm 18.9$ deg/m, is not significant.

The experiment was underpowered before it began: the expected effect of a 0.03 trim step is roughly 15 deg/m against a ± 19 uncertainty, requiring about 19 legs per point rather than six. Pooling all twelve legs gives a real leftward bias of $+26.7 \pm 9.6$ deg/m, but the leg-to-leg scatter (sd ≈ 32) exceeds it, and no single trim value corrects a variance. The trim was returned to its prior value and the issue is left open as mechanical rather than parametric. The methodological lesson — run the power calculation before spending experimental time, not after — is recorded because it was learned the expensive way.

8.5 Retracted conclusions

Three conclusions reached during development were later overturned by evidence. They are reported because the sequence is instructive.

1. “*The robot cannot pivot; wheel slip is 98%.*” Derived from raw IMU angular velocity, which this driver under-reports by a factor of exactly $180/\pi$. The corrected figure showed agreement within 5%. Caught by the operator observing that the robot was visibly spinning.
2. “*The gyro is the untrustworthy source.*” After a corridor run in which the gyro disagreed with both SLAM and odometry, the gyro was judged the outlier. Controlled turn tests against direct observation showed the gyro correct in both, and wheel odometry wrong by up to $5\times$.
3. “*Heatmapping is blocked; the variation is purely temporal.*” True of runs in which the robot swept the space once, confounding position with time. In a run traversing back and forth, $r(x, t) = 0.12$ and detrending *raised* temperature’s spatial correlation from 0.35 to 0.61.

In each case the error was corrected by an independent reference — most often the operator’s direct observation — rather than by further analysis of the same data. That is the practical lesson: on a platform without ground truth, a second opinion of a different kind is worth more than a better statistic.

8.6 Metric traps

- **Waypoints reached is not coverage.** A run can reach every waypoint of a bad plan.

- **Mapped area is not map quality.** The ablation shows the broken configuration scoring marginally *higher* on free area.
- **Dwell time biases spatial statistics.** Where the robot pauses contributes disproportionately many samples.
- **Single runs are not comparable** in this environment. A four-run baseline in one simulated world spanned 18 coverage points, and at least three claims in this project were initially made, then withdrawn, on single-run evidence.

9 Discussion

The system does what was intended in the geometric domain: it explores unsupervised, produces navigation-grade occupancy maps of a real corridor, crosses that corridor repeatably without contact, and detects a change in the environment’s configuration — an opened door — with high signal-to-noise and full reversibility. In simulation it recovers a known environmental field almost exactly while correctly failing to recover fields that do not exist.

The environmental claim on hardware is weaker and should be stated as such. Temperature and humidity show spatial structure once the sensor’s self-heating is removed; TVOC and eCO₂ do not, in a corridor where they should not. The one strongly-powered environmental result — ventilation detection through the sensor’s thermal balance — does not require mobility at all. The honest summary is that this work demonstrates a mobile platform that maps geometry well and carries environmental sensors reliably, and that the case for *mobility* in the environmental measurement remains to be made by a spatial result the present sampling strategy cannot deliver.

That gap has an identified cause and a concrete remedy. The patrol drives between waypoints and pauses at each; for environmental sampling it should traverse continuously. This is a clean separation of objectives that share a vehicle, and it is the single change most likely to convert the environmental side of this system from plausible to demonstrated.

A broader point concerns evaluation on platforms without ground truth. The central defect in this project persisted for weeks not because it was subtle in its effects — it cost 4.5 m of localization error — but because every available metric was computed from the estimate that was itself wrong. The map looked right; the area was right; the scan matched itself. Only an external reference could contradict it, and on hardware the only external reference available was a person looking at the robot. We suggest that low-cost mapping work should budget explicitly for such references: a tape measure, a marked floor, a person watching a commanded motion.

10 Limitations

1. **One environment, one robot.** All hardware results come from a single 4.6×1.0 m corridor and one DB21. No room of sufficient size was available. Room-scale results are simulated only.
2. **Absolute temperature is not trustworthy.** The sensor reads 32–35°C in a room that is not at that temperature, reflecting board self-heating. Only relative spatial structure and rate-of-change are claimed; the sensor was never checked against a reference thermometer.

3. **Air-quality mapping is not demonstrated** on hardware. TVOC and eCO₂ showed no spatial structure in the corridor.
4. **The spatial thermal experiment is underpowered**, with effect sizes at the level of the measurement noise once dwell samples are excluded.
5. **map $\rightarrow$ odom excursions remain unexplained**. They reached 170–180° on runs whose maps and navigation were otherwise good, and varied widely between runs (33.9°, 107.9°, 179.7°). A stable frame offset in an end-to-end symmetric corridor is a plausible explanation and is not verified.
6. **Ground truth is a tape measure**, not a survey. Consistent with a navigation-grade rather than metrology-grade claim.
7. **Simulation does not reproduce wheel slip** faithfully, which is why the gyro-fusion change could not be ablated there and rests on hardware measurement against operator observation.
8. **Repeat count is small**. Four clean traverses post-fix, three door-experiment runs, two repeats per ablation arm.

11 Future Work

Continuous-traverse sampling. Replace waypoint-and-pause motion with constant-speed traversal for the sensing pass, and re-run the spatial thermal experiment. This is the highest-value next step and directly targets the identified cause of the failed spatial measurement.

The ventilation model. The motivating application: given a spatial and temporal description of temperature and CO₂, estimate how long a window must remain open to clear accumulated CO₂, and the associated heat loss. The present work delivers the instrument; the model requires a room, a reference-grade CO₂ sensor and a controlled occupancy protocol.

Sensor extrinsics verification as a startup check. The nudge test that revealed the reversed LiDAR — command a small forward motion, observe which bearings’ ranges decrease — is cheap and automatable, and would have caught the defect on day one.

Reference instrumentation. A calibrated thermometer and CO₂ meter would convert relative claims into absolute ones, and a marked floor would give hardware localization an external reference.

12 Conclusion

A Duckiebot DB21 with an added planar LiDAR and environmental sensor was built into an autonomous scanner that explores an indoor space, maps it, and records geotagged environmental measurements. In simulation across seven environments the system recovers a known environmental field at $r = 0.996$ while correctly recovering nothing from structureless controls, and coverage-aware re-planning improves sensed coverage by up to 39% over coverage-blind planning. On hardware it traverses a real corridor repeatably, produces navigation-grade maps consistent with tape-measured

ground truth, and detects an opened door both geometrically — a 1.7 m mapped wall excursion, reversible to 2 cm — and through the effect of ventilation on its own thermal balance.

The report’s most transferable content is negative. A 180° sensor-extrinsics error survived weeks of development because the corridor available for testing conceals it and because every map-quality metric in use is computed from the estimate that is itself wrong; a controlled ablation shows it costing 4.5 m of localization error while leaving mapped area unchanged. Wheel odometry was found to fabricate rotation the robot never performed. Two sensors failed silently while reporting healthy. Two pre-registered predictions failed and are reported as failures. Three conclusions reached during development were retracted on later evidence, in each case corrected by an independent observation rather than by more analysis of the same data.

A Duckiebot room scanner is technically credible, as the literature suggested. What this work adds is a measured account of the conditions under which such a system is trustworthy, and of the specific ways in which it can appear to work while being substantially wrong.

13 Code and Data Availability

Everything described in this report — the ROS workspace, every node, the simulation worlds, the analysis tooling, the raw experimental data and the papers — is available at:

<https://gitlab.hua.gr/mobile-sensor-unit>

The experimental archive is organised so that runs made under *different system configurations are not mixed*, because the sensor-extrinsics correction described in Section 8 makes runs either side of it incomparable. Directories are prefixed accordingly: runs carrying both fixes, runs predating them (retained as failure evidence only), and calibration procedures.

Each hardware run includes the occupancy grid and its metadata, the geotagged sensor CSV, the SLAM health trace, the three-way heading comparison, the per-waypoint validation verdicts and the full console log. Each simulation experiment includes its summary `results.csv`; the ablation additionally includes Gazebo ground-truth pose traces, which are the only localization reference in the project not derived from the estimate being assessed.

The pre-registered predictions for the experiment in Section 7 were committed to version control before any of that data was collected, and are included alongside the outcomes.

The analysis tools require only `numpy`, so every figure and table here can be regenerated from the raw data without installing a scientific Python stack.

A Supplementary Tables

The two literature-survey tables referenced from Section 2 are collected here rather than inline, since both are landscape-oriented and each spans more than a page.

Table 10: Duckiebot evidence ranked by directness to the room-scanning question.

Study / project	Evidence tier	Physical sensing	Mapping / localization output	Room-scan relevance	Assessment
Paull et al. 2017 [1]	A	Monocular camera; onboard compute	Localization in a global Duckietown map; navigation	Medium	Foundational proof of map-based autonomy, but not unknown-room geometry mapping.
EKF-SLAM for Duckiebots [3]	B	DB21-J4; wheel encoders + camera AprilTags	Joint robot-pose and landmark-position estimate	Medium	Real SLAM formulation; maps fiducial landmarks rather than walls/furniture.
Anatidaephilia cSLAM [4]	B	Duckiebot cameras + external watchtowers + AprilTags	Centralized city/tag map and robot localization	Low–Medium	Mapping is structured and infrastructure-assisted, not free-form room scanning.
Magiera et al. 2024 [28]	A	Multiple external cameras + AprilTags; Duckietown testbed	Stitched top-down views and multi-robot pose localization	Low–Medium	Peer-reviewed localization/mapping-adjacent evidence; not onboard room scanning.
KnightCar [5]	B	Duckiebot-derived robot + 2D LiDAR	2D particle-filter SLAM, EKF-SLAM, AMCL, A* navigation	High	Closest documented Duckiebot precedent for a mobile 2D room-mapping stack; not peer reviewed.
Northeastern Duckiebot project [6]	C	Monocular camera + SlamTech 2D LiDAR	Hector SLAM map building and robot localization	High	Direct LiDAR-SLAM evidence on a Duckiebot, but archival evidence is a project profile rather than paper.
Li & Okhrin 2023/2024 [7, 8]	A	DB21 + standard sensors + RPLIDAR A1M8	Driving perception/control; not SLAM	Medium	Strong peer-reviewed hardware-feasibility evidence for adding a scanning LiDAR to DB21.
Informal recent RTAB-Map-style Duckiebot projects	C	Camera/odometry or added sensors	2D map/localization demonstrations	Medium	Support feasibility, but publication quality is insufficient to anchor the scientific conclusion.

Table 11: Major Duckiebot research uses with representative evidence.

Application	Typical sensors / inputs	Representative methods	Representative source	What the Duckiebot contributes
Lane following / lane keeping	Monocular camera; sometimes encoders	Classical CV + PID; CNN; RL	Paull et al. [1]; Gupta & Easwaran [21]; Kalapos et al. [23]	Low-cost physical closed-loop autonomy and embedded execution.
Obstacle detection and avoidance	Monocular camera; optional range sensors	Inverse perspective mapping; learned policies	Nubert et al. [29]; Kalapos et al. [23]	Real-time perception under severe compute limits.
Traffic-sign recognition	Fisheye camera	Color/shape vision; classification	Thiang et al. [32]	Controlled physical testbed for perception-to-action behavior.
Localization and state estimation	Encoders, camera, AprilTags, IMU	EKF; sensor fusion; PnP	EKF-SLAM project [3]; Duckietown manuals [2]	Reproducible study of uncertainty and sensor fusion.
SLAM / structured mapping	Camera, AprilTags, odometry; sometimes LiDAR	EKF-SLAM; centralized SLAM; LiDAR SLAM	[3]–[6]	Low-cost platform for comparing mapping architectures.
Path planning and navigation	Map + pose estimate	Dijkstra/A*; map-based navigation	Paull et al. [1]; KnightCar [5]	Integration of perception, planning, and control.
Reinforcement learning	Camera images; simulator observations	DQN, PPO, MAPPO	Almási et al. [18]; Kalapos et al. [23]; Candela et al. [20]	Physical validation of learned policies and reality-gap studies.
Imitation learning	Camera images; expert driving logs	Behavior cloning / imitation learning	Lőrincz et al. [27]; Sazanovich et al. [30]	Standardized real/sim data and inexpensive policy deployment.
Sim-to-real transfer	Synthetic + real camera observations	Domain randomization; domain adaptation	Tim et al. [33]; Li & Okhrin [7, 8]	Controlled comparison between simulator and real robot.
Multi-agent coordination	Multiple Duckiebots; cameras; shared environment	MAPPO; rule-based coordination; fleet tasks	Candela et al. [20]; Zilly et al. [35]	Scalable physical multi-robot experiments.
Safety / OOD detection	Camera and embedded ML outputs	VAE/OOD detection; emergency stop	Yuhás et al. [34]	Evaluation of embedded AI failure detection on real hardware.
Automatic calibration	Camera; wheels; fiducial-marked area	Autonomous camera and wheel calibration	Krinkin et al. [25]	Repeatable calibration procedures for fleets of small robots.
Multi-camera visual localization	External cameras + AprilTags	Homography, stitching, top-down reprojection	Magiera et al. [28]	Flexible global localization for multi-robot testbeds.
Traffic and intersection research	Duckiebots + traffic-light infrastructure	Traffic control, interaction policies, evaluation	AI-DO / Duckietown ecosystem [31, 35]	Miniature city enabling networked autonomy experiments.
Reproducibility and benchmarking	Simulation, logs, remote physical robots	Standard interfaces and automated evaluation	Tani et al. [31]	Low-cost standardized hardware for repeatable robotics experiments.
Autonomy education	Camera, encoders, IMU, ToF; ROS	Kinematics, vision, filtering, planning, ML	Paull et al. [1]; official learning materials [2]	One platform spans fundamentals through advanced embodied AI.
Car following and overtaking	Camera; RPLIDAR on DB21 in advanced setup	DRL + perception abstraction	Li & Okhrin [7, 8]	Demonstrates extensible sensing and more complex driving behaviors.

References

- [1] L. Paull *et al.*, “Duckietown: An Open, Inexpensive and Flexible Platform for Autonomy Education and Research,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1497–1504, 2017. <https://doi.org/10.1109/ICRA.2017.7989179>
- [2] Duckietown, “Duckiebot Models and Hardware/Learning-Experience Documentation,” Official Duckietown manuals. <https://docs.duckietown.com/ente/opmanual-duckiebot/07-further-reading/duckiebot-configurations-explained.html>
- [3] A. H. Zamani, L. Oest O’Leary, and K. Lessard, “Extended Kalman Filter (EKF) SLAM for Duckiebots,” University of Montreal / Duckietown project page, 2025. <https://duckietown.com/extended-kalman-filter-slam-for-duckiebots/>
- [4] R. Suri, A. Camus, F. Milano, B. Kuan, and A. Petrov, “Anatidaephilia: centralized city-based SLAM (cSLAM),” Duckietown project page. <https://duckietown.com/anatidaephilia-centralized-city-based-slam-cslam/>
- [5] P. F. Kung, “KnightCar: low-cost ROS autonomous platform,” Primary author project page. <https://kungfrank.github.io/>
- [6] S. Xu, “Duckiebot Autonomous Mobile Robot,” Northeastern University project record, 2020. https://github.com/seanxu889/CS5335_RSS
- [7] D. Li, and O. Okhrin, “Vision-Based DRL Autonomous Driving Agent with Sim2Real Transfer,” in *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 866–873, 2023. <https://doi.org/10.1109/ITSC57777.2023.10422677>
- [8] D. Li, and O. Okhrin, “A platform-agnostic deep reinforcement learning framework for effective Sim2Real transfer towards autonomous driving,” *Communications Engineering*, vol. 3, pp. 147, 2024. <https://doi.org/10.1038/s44172-024-00292-3>
- [9] G. Grisetti, C. Stachniss, and W. Burgard, “Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters,” *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007. <https://doi.org/10.1109/TR0.2006.889486>
- [10] B. Yamauchi, “A frontier-based approach for autonomous exploration,” in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation*, pp. 146–151, 1997. <https://doi.org/10.1109/CIRA.1997.613851>
- [11] A. Elfes, “Using Occupancy Grids for Mobile Robot Perception and Navigation,” *Computer*, vol. 22, no. 6, pp. 46–57, 1989. <https://doi.org/10.1109/2.30720>
- [12] S. Kohlbrecher, O. von Stryk, J. Meyer, and U. Klingauf, “A flexible and scalable SLAM system with full 3D motion estimation,” in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pp. 155–160, 2011. <https://doi.org/10.1109/SSRR.2011.6106777>
- [13] D. Fox, W. Burgard, and S. Thrun, “The Dynamic Window Approach to Collision Avoidance,” *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997. <https://doi.org/10.1109/100.580977>
- [14] M. Quigley *et al.*, “ROS: an open-source Robot Operating System,” in *ICRA Workshop on Open Source Software*, 2009.

- [15] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, “The Office Marathon: Robust Navigation in an Indoor Office Environment,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2010. <https://doi.org/10.1109/ROBOT.2010.5509725>
- [16] H. Choset, “Coverage for Robotics – A Survey of Recent Results,” *Annals of Mathematics and Artificial Intelligence*, vol. 31, pp. 113–126, 2001. <https://doi.org/10.1023/A:1016639210559>
- [17] D. Shepard, “A Two-Dimensional Interpolation Function for Irregularly-Spaced Data,” in *Proceedings of the 23rd ACM National Conference*, pp. 517–524, 1968. <https://doi.org/10.1145/800186.810616>
- [18] P. Almási, R. Moni, and B. Gyires-Tóth, “Robust Reinforcement Learning-based Autonomous Driving Agent for Simulation and Real World”, in *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2020. <https://doi.org/10.1109/IJCNN48605.2020.9207497>
- [19] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, “ORB-SLAM3: An Accurate Open-Source Library for Visual, Visual-Inertial, and Multimap SLAM”, *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021. <https://doi.org/10.1109/TR0.2021.3075644>
- [20] E. Candela, L. Parada, L. Marques, T. A. Georgescu, Y. Demiris, and P. Angeloudis, “Transferring Multi-Agent Reinforcement Learning Policies for Autonomous Driving using Sim-to-Real”, in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8814–8820, 2022. <https://doi.org/10.1109/IROS47612.2022.9981319>
- [21] A. Gupta and A. Easwaran, “A Low-Cost Lane-Following Algorithm for Cyber-Physical Robots”, arXiv preprint arXiv:2208.10765, 2022. <https://doi.org/10.48550/arXiv.2208.10765>
- [22] W. Hess, D. Kohler, H. Rapp, and D. Andor, “Real-Time Loop Closure in 2D LIDAR SLAM”, in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1271–1278, 2016. <https://doi.org/10.1109/ICRA.2016.7487258>
- [23] A. Kalapos, C. Górr, R. Moni, and I. Harmati, “Vision-based reinforcement learning for lane-tracking control”, *Acta IMEKO*, vol. 10, no. 3, pp. 7–14, 2021. https://doi.org/10.21014/acta_imeko.v10i3.1020
- [24] G. Karalekas, S. Vologiannidis, and J. Kalomiros, “EUROPA: A Case Study for Teaching Sensors, Data Acquisition and Robotics via a ROS-Based Educational Robot”, *Sensors*, vol. 20, no. 9, pp. 2469, 2020. <https://doi.org/10.3390/s20092469>
- [25] K. Krinkin, K. Chayka, A. Filatov, and A. Filatov, “Autonomous Wheels And Camera Calibration In Duckietown Project”, *Procedia Computer Science*, vol. 186, pp. 169–176, 2021. <https://doi.org/10.1016/j.procs.2021.04.135>
- [26] M. Labbé and F. Michaud, “RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation”, *Journal of Field Robotics*, 2019. <https://doi.org/10.1002/rob.21831>
- [27] Z. Lőrincz, M. Szemenyei, and R. Moni, “Imitation Learning for Generalizable Self-driving Policy with Sim-to-real Transfer”, arXiv preprint arXiv:2206.10797, 2022. <https://doi.org/10.48550/arXiv.2206.10797>
- [28] A. M. Magiera, M. Długosz, and P. Skruch, “Multi-camera multi-robot visual localization system”, in *2024 28th International Conference on Methods and Models in Automation and Robotics (MMAR)*, pp. 375–380, 2024. <https://doi.org/10.1109/MMAR62187.2024.10680813>

- [29] J. Nubert, N. Funk, F. Meier, and F. Oehler, “Developing a Purely Visual Based Obstacle Detection using Inverse Perspective Mapping”, arXiv preprint arXiv:1809.01268, 2018. <https://doi.org/10.48550/arXiv.1809.01268>
- [30] M. Sazanovich, K. Chaika, K. Krinkin, and A. Shpilman, “Imitation Learning Approach for AI Driving Olympics Trained on Real-world and Simulation Data Simultaneously”, in *Workshop on AI for Autonomous Driving, ICML 2020*, 2020. <https://doi.org/10.48550/arXiv.2007.03514>
- [31] J. Tani, A. F. Daniele, G. Bernasconi, A. Camus, A. Petrov, A. Courchesne, B. Mehta, R. Suri, T. Zaluska, M. R. Walter, E. Frazzoli, L. Paull, and A. Censi, “Integrated Benchmarking and Design for Reproducible and Accessible Evaluation of Robotic Agents”, in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6229–6236, 2020. <https://doi.org/10.1109/IROS45743.2020.9341677>
- [32] Thiang St, G. R. Purwanto, P. Santoso, and H. Khoswanto, “Autonomous mobile robot development based on duckietown platform for recognizing and following the traffic sign”, *Journal of Physics: Conference Series*, vol. 1921, pp. 012065, 2021. <https://doi.org/10.1088/1742-6596/1921/1/012065>
- [33] M. Tim, M. Szemenyei, and R. Moni, “Simulation to Real Domain Adaptation for Lane Segmentation”, in *2020 23rd International Symposium on Measurement and Control in Robotics (ISMCR)*, pp. 1–6, 2020. <https://doi.org/10.1109/ISMCR51255.2020.9263406>
- [34] M. Yuhas, Y. Feng, D. J. X. Ng, Z. Rahiminasab, and A. Easwaran, “Embedded Out-of-Distribution Detection on an Autonomous Robot Platform”, in *Proceedings of the Workshop on Design Automation for CPS and IoT (DESTION '21)*, pp. 13–18, 2021. <https://doi.org/10.1145/3445034.3460509>
- [35] J. Zilly, J. Tani, B. Considine, B. Mehta, A. F. Daniele, M. Diaz, G. Bernasconi, C. Ruch, J. Hakenberg, F. Golemo, A. K. Bowser, M. R. Walter, R. Hristov, S. Mallya, E. Frazzoli, A. Censi, and L. Paull, “The AI Driving Olympics at NeurIPS 2018”, in *The NeurIPS '18 Competition*, R. Escalera and R. Herbrich, Eds., Springer, Cham, pp. 37–68, 2020. https://doi.org/10.1007/978-3-030-29135-8_3
- [36] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. MIT Press, 2005.