Wrexham University Research Online

**Conference Paper**

# Edge AI-Enabled Person Detection and LiDAR-Only SLAM for a Low-Cost SAR Robot

Strang, J., Sharp, A., and Durieux, O.

This is a paper presented at 2026 10th International Conference on Mechatronics Engineering (ICOM).

Copyright of the author(s). Reproduced here with their permission and the permission of the conference organisers.

This accepted manuscript is licensed under a Creative Commons Attribution 4.0 International (CC BY 4.0) licence: https://creativecommons.org/licenses/by/4.0/
The Version of Record can be found here: https://ieeexplore.ieee.org/document/11726303

**Recommended citation:**

Strang, J., Sharp, A. and Durieux, O. (2026) 'Edge AI-enabled person detection and LiDAR-only SLAM for a low-cost SAR robot', *2026 10th International Conference on Mechatronics Engineering (ICOM)*, Malaysia, pp. 1–7. Available at: https://doi.org/10.1109/ICOM71159.2026.11726303

# Edge AI-Enabled Person Detection and LiDAR-Only SLAM for a Low-Cost SAR Robot

Jörg Strang
*Wrexham University*
Wrexham, UK

Andrew Sharp
*Wrexham University*
Wrexham, UK

Olivier Durieux
*Wrexham University*
Wrexham, UK

***Abstract*—Autonomous search and rescue (SAR) robots offer a way to reduce risk to human search teams during disaster first response. However, commercial platforms are too costly for widespread deployment, while low-cost alternatives fail due to wheel odometry errors on debris and insufficient computing power for parallel navigation and person detection. This paper presents a £431.68 prototype built on a Raspberry Pi 5, using only commercial off-the-shelf components and open-source software. Two design choices resolve the computational bottleneck. First, the SLAM-Toolbox is configured for odometry-less operation, using LiDAR scan matching alone for position estimation. Second, YOLOv8n inference is offloaded to the Sony IMX500 on-sensor Digital Signal Processor, freeing the CPU for simultaneous SLAM and Nav2 navigation. A Foxglove operator dashboard integrates the live map, detection camera, and battery telemetry into a single interface. Over ten indoor test runs, the system achieved a 100% success rate in six consecutive runs after parameter tuning (runs 4–9), with an average Time to Detection of 2–3 minutes in a 55 $m^2$ three-room course.**

***Index Terms*—search and rescue robotics, autonomous navigation, LiDAR SLAM, edge AI inference, ROS 2, low-cost prototype**

## I. Introduction

Natural disasters and urban building collapses are time-critical events. The survival probability of trapped victims falls sharply after the first 72 hours [1], [2]. Rubble fields are often too unstable for immediate entry by human search teams or rescue dogs, placing first responders at serious personal risk [3]. Autonomous robots can enter spaces that are too narrow or too dangerous for people, operating during the critical early phase when human access is restricted [4], [5].

Despite decades of research, two barriers prevent the widespread deployment of autonomous SAR robots in civil protection [6]. First, high-performance platforms such as Boston Dynamics Spot offer strong autonomy but cost tens of thousands of pounds, making them impractical as expendable units [6], [7]. Second, low-cost open-source platforms lack the computing power required to run simultaneous localisation and mapping (SLAM) alongside real-time person detection on the same hardware [8]. These low-cost designs also depend on wheel encoders for odometry, which fail on loose debris due to wheel slip [9], [10]. Together, these limitations create a gap between what civil protection organisations need and what current technology can deliver.

This paper presents an integrated low-cost implementation and experimental evaluation of existing open-source robotics and edge-AI tools for an indoor SAR prototype. The system runs on a Raspberry Pi 5 and uses only commercial off-the-shelf (COTS) components, with a total cost of £431.68. The work makes three practical contributions. First, the SLAM-Toolbox is configured to operate without wheel encoders, using LiDAR scan matching alone for position estimation. Second, YOLOv8n person detection is offloaded to the Sony IMX500 Digital Signal Processor within the camera module, freeing the CPU entirely for navigation. Third, a Foxglove operator dashboard integrates the live SLAM map, detection camera, and battery telemetry into a single interface accessible from a standard laptop over a local Wi-Fi access point, reducing operator cognitive load [11].

The system was validated in a 55 $m^2$ three-room indoor test course over ten runs. After parameter tuning, it achieved a 100% success rate in runs 4–9 with an average Time to Detection of 2–3 minutes.

## II. Related Work

**SAR robotics and open-source platforms.** The deployment of Inuktun and Talon robots at the World Trade Centre in 2001 marked the beginning of dedicated SAR robotics research [3]. Those platforms were teleoperated and offered no autonomous capabilities, producing high cognitive load for operators and frequent communication failures [4]. Research since then has moved towards autonomy, culminating in large multi-robot frameworks such as the EU ICARUS project, which coordinated aerial, ground, and aquatic units through a shared command interface [5]. These high-end systems offer strong autonomy but remain too costly for widespread use in civil protection [6], [7]. Nwankwo *et al.* [12] demonstrated that a ROS-based open-source mobile robot (ROMR) can match far more expensive commercial platforms for under $1,500, showing that COTS components and open-source software represent a viable strategy for cost reduction.

**SLAM without wheel odometry.** Most low-cost SAR platforms still rely on wheel encoders for odometry. On loose debris, wheel slip introduces position errors that cause SLAM maps to tear and produce ghost walls [9], [13]. Zhang *et al.* [10] show that graph-based, LiDAR-only SLAM using scan matching can achieve accurate room geometry in structured indoor environments without any encoder data. This performance depends on the presence of sufficient distinctive features in the environment, such as walls and corners. However, no validated study demonstrates this approach on a sub-£500 platform performing concurrent navigation and real-time detection.

**Edge AI for constrained hardware.** Running neural network inference on a single-board computer CPU causes resource contention with navigation algorithms. Vaghela *et al.* [8] show that SLAM performance degrades significantly when inference runs on the same CPU as the navigation

TABLE I
SELECTED HARDWARE COMPONENTS

| Component | Selected | Alternative(s) | Rationale |
|---|---|---|---|
| Control unit | Raspberry Pi 5 | RPi 4, Jetson Nano | Higher CPU performance; ROS 2 Jazzy support [18], [19] |
| LiDAR | RPLIDAR C1 | RPLIDAR A-series | Compact, stable data rate, low cost [20] |
| Camera | RPi AI camera | RPi camera module 3 | Hardware acceleration (Sony IMX500) [17], [21] |
| Drive unit | TT-motor 1:90 | TT-motor 1:48 | Higher torque; precise PWM control [22] |
| Power supply | Waveshare UPS 3S | Power bank | Dual-rail output; charge management [23] |
| Collision sensor | US-100 ultasonic | HC-SR04 | 3.3V logic; no level converter needed [24] |

stack. Recent surveys confirm that edge inference for robotics is an active research area, covering general robotic edge-computing architectures [14], resource-constrained embedded vision pipelines using TinyML [15], and benchmarking of object detection models on low-compute edge system-on-chip devices [16]. Offloading inference to a dedicated hardware accelerator resolves this contention, but existing approaches rely on separate GPU boards or cloud servers, which add cost and system complexity [17]. The Sony IMX500 on-sensor DSP offers hardware-level inference offloading within a single low-cost camera module, but its integration with a full ROS 2 SLAM and Nav2 stack has not previously been demonstrated.

**Operator cognitive load.** Murphy [11] identifies cognitive overload as a leading cause of mission failure in SAR robotics, independent of hardware quality. Translating raw sensor data into a unified tactical map significantly reduces the training burden on first responders. However, practical guidance on achieving this with open-source tools on low-cost hardware remains limited [4].

This paper fills all three gaps within a single integrated system: LiDAR-only SLAM without wheel odometry, on-sensor YOLOv8n inference offloaded to the IMX500 DSP, and a unified Foxglove operator dashboard, all running on a £431.68 Raspberry Pi 5 platform.

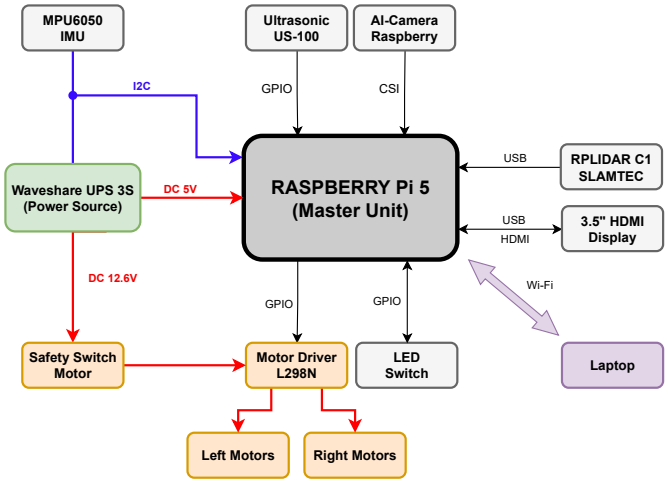



Fig. 1. Hardware block diagram: connections and data flow between components.



Fig. 2. Assembled SAR prototype: front view (left) and rear view (right).

## III. SYSTEM DESIGN

### A. Hardware Architecture

The prototype is a 4WD ground robot built on a three-level, additively manufactured chassis (total mass: 1.35 kg), designed in Autodesk Fusion 360 and printed with a Bambu Lab X1E. The layered layout separates functional subsystems and lowers the centre of gravity for stability on uneven surfaces. Level 1 (bottom) holds the motors and power module. Level 2 (middle) contains the Raspberry Pi 5 and associated electronics, with ventilation slots to prevent thermal throttling. Level 3 (top) supports the LiDAR sensor and a 3.5-inch HDMI operator display. Individual levels are connected by machine screws, enabling tool-free disassembly for maintenance.

The Raspberry Pi 5 (2.4 GHz quad-core, 8 GB RAM) acts as the central processing unit [18]. Table I lists the selected components alongside the rejected alternatives. The Raspberry Pi 4 was excluded due to its lower clock speed, and the NVIDIA Jetson Nano due to its higher cost and limited support for the ROS 2 Jazzy distribution [25].

The SLAMTEC RPLIDAR C1 provides 360° horizontal scanning via USB [20]. The MPU-6050 IMU supplies raw gyroscope and accelerometer data at 20 Hz via the $I^2C$ bus [26]. The Raspberry Pi AI Camera connects via the Camera Serial Interface (CSI) port and runs YOLOv8n inference directly on the Sony IMX500 Digital Signal Processor (DSP), keeping the host CPU entirely free for navigation [17].

The US-100 ultrasonic sensor operates at 3.3 V logic, requires no level converter, and detects obstacles below the LiDAR scanning plane at a ground clearance of 15 mm [24].

Power is supplied by the Waveshare UPS 3S module with three 3000 mAh Li-ion cells [23]. It provides DC 5 V for logic via USB-C and DC 12.6 V for the drive subsystem via the XH2.54 port. An emergency switch in the 12.6 V line allows the motor circuit to be cut independently of the logic subsystem. Total power consumption under full load is 24.05 W (13.97 W logic, 10.08 W drive). With a nominal battery voltage of $U_{\text{nom}} = 11.1$ V, capacity $Q = 3$ Ah, and DC-DC converter efficiency $\eta = 0.85$, the theoretical battery runtime is:

$$t = \frac{U_{\text{nom}} \cdot Q}{P_{\text{total}}/\eta} = \frac{11.1 \times 3}{24.05/0.85} \approx 71 \text{ min} \qquad (1)$$

This exceeds the 15-minute mission target by a factor of four. A software interrupt at 9.5 V prevents deep discharge. Fig. 1

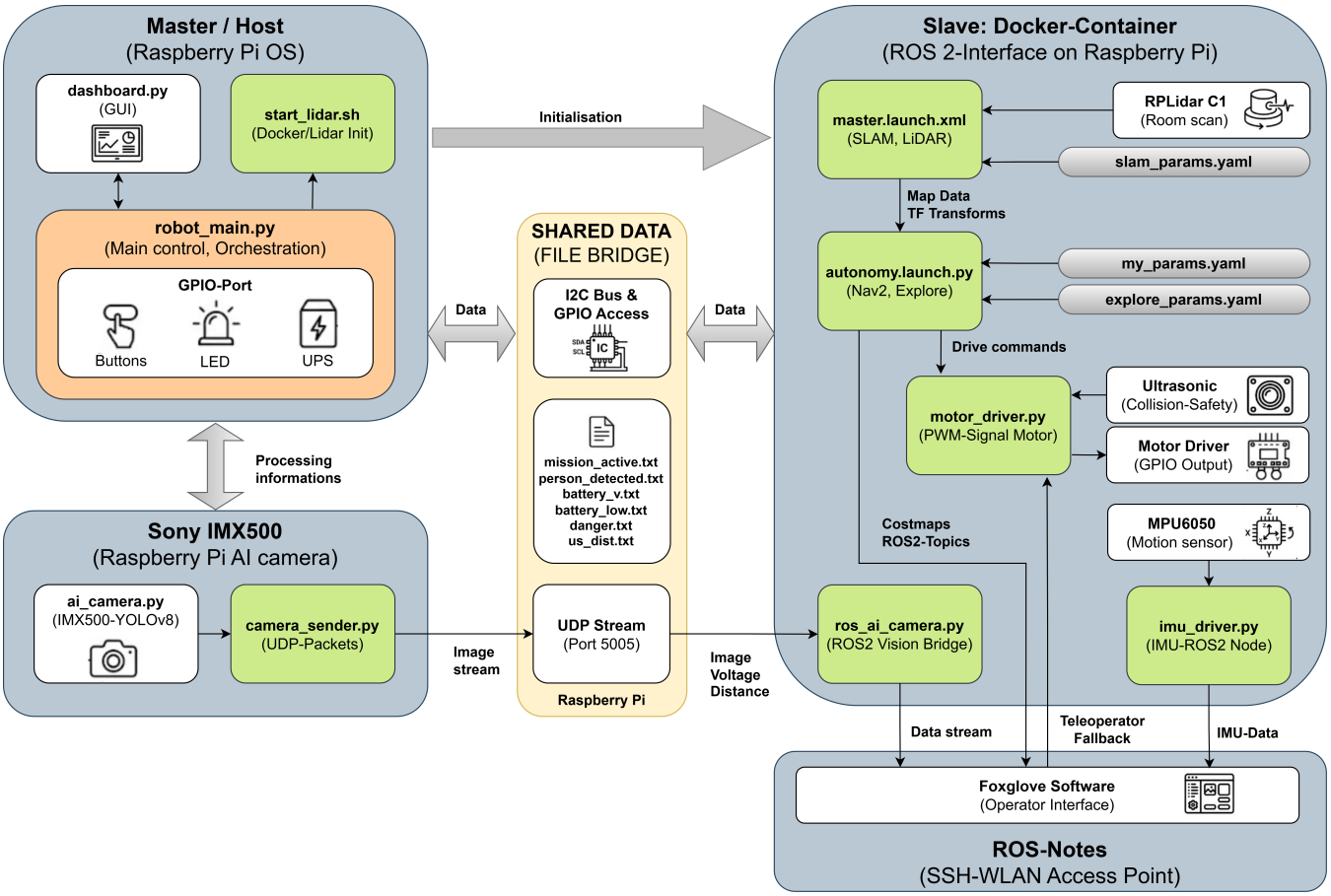



Fig. 3. Software architecture: host OS communicates via a shared file bridge with the ROS 2 Docker container and separately with the IMX500 AI camera.

shows the full hardware block diagram and Fig. 2 shows the assembled prototype.

### B. Software Architecture

The software uses a hierarchical master-slave model. The host operating system and an isolated Docker container run in parallel on the Raspberry Pi 5 (Fig. 3). Docker isolation prevents library conflicts between the proprietary IMX500 camera drivers and the ROS 2 stack. ROS 2 was selected over ROS 1 because it uses the Data Distribution Service (DDS) for decentralised, peer-to-peer communication [27]. Unlike the ROS 1 central server model, where a single node crash can paralyse the entire network, DDS provides quality-of-service control that prioritises safety-critical messages such as emergency stop commands, even under high CPU load.

**Host level.** The `robot_main.py` script manages the Docker container lifecycle and acts as the system watchdog. It reads GPIO button inputs, controls the RGB-LED status indicator, and monitors battery voltage via $I^2C$. When voltage falls below 9.5 V, it shuts down the motors immediately. AI camera inference also runs at host level to avoid driver instability inside the container.

**Container level (ROS 2 Jazzy).** Start-up follows a two-stage sequence. The first stage (`master.launch.xml`) initialises the RPLIDAR C1 driver, the MPU-6050 IMU driver, the SLAM-Toolbox in asynchronous mode, and the Foxglove WebSocket bridge. The second stage (`autonomy.launch.py`) activates the Nav2 stack and the `explore-lite` frontier exploration node. Inter-process communication uses two mechanisms: system state flags (e.g., `person_found.txt`) pass through a shared file bridge, while the live video stream uses a local UDP socket on port 5005, keeping high-frequency image data off the ROS 2 topic network and reducing CPU load.

**Priority chain.** Motor commands follow a strict four-level hierarchy: (1) battery protection shuts down the motors if voltage drops below 9.5 V; (2) the US-100 sensor blocks forward motion if an obstacle is within 15 cm, while still permitting Nav2 recovery manoeuvres such as reversing and turning; (3) a confirmed person detection event immediately terminates `explore-lite` and stops the robot; (4) Nav2 autonomous navigation has the lowest priority and can be overridden at any time by a Foxglove virtual teleoperator publishing directly to `/cmd_vel`.

**Transform Tree and coordinate alignment.** The ROS 2 Transform Tree (TF) defines the spatial relationship between every sensor and the robot body. All sensor-to-chassis offsets, for the LiDAR (`laser_link`), IMU (`imu_link`), camera (`camera_link`), and ultrasonic sensor (`ultrasonic_link`), are defined as static transforms in `master.launch.xml` (Fig. 4). This means their positions are set once at start-up from the CAD model and never recomputed, adding no runtime CPU overhead. The critical transform is `odom` → `base_link`: without wheel encoders, this was set to a static zero transform. All motion estimation is therefore performed dynamically by

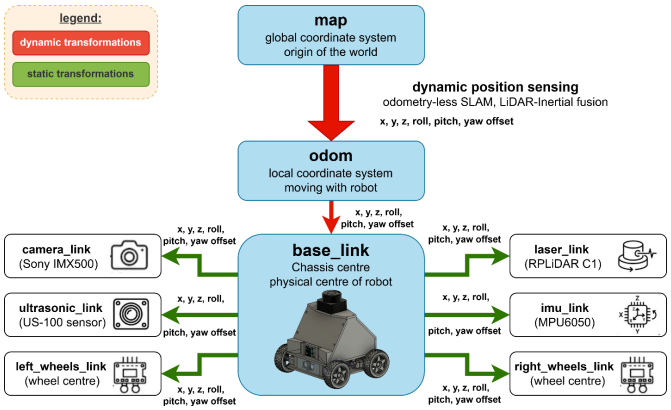



Fig. 4. SAR robot Transform Tree (TF) hierarchy. Static transforms (green) define all sensor-to-chassis offsets from the CAD model. The dynamic transform (red) from `map` to `odom` is estimated entirely by the SLAM-Toolbox via LiDAR scan matching; the `odom` → `base_link` transform is set to zero, replacing wheel odometry.

the SLAM-Toolbox using LiDAR scan matching alone. A `transform_tolerance` of 2.0 s was set to absorb short-term latency spikes between the SLAM-Toolbox and Nav2 without interrupting navigation.

**Key design decisions.** Four parameter choices were critical for stable operation and represent the main engineering contribution of this work. (1) No wheel encoders: debris causes wheel slip, making encoder odometry unreliable in SAR scenarios [9]. Using LiDAR scan matching for position estimation removes this dependency by design, since no wheel data is used for localisation. This study did not test the design on physical debris; see Section V for the test conditions used. (2) Edge AI offloading: running YOLOv8n on the host CPU alongside SLAM and Nav2 caused ROS 2 buffer overflows; moving inference to the IMX500 DSP eliminated all timeouts [8], [17]. (3) Costmap resolution reduced from 5 cm to 10 cm: the finer grid generated four times the memory load and overflowed the ROS 2 message queue; a 10 cm resolution resolved this issue without any loss of doorway detection accuracy. (4) Rotation speed limited at both hardware and software levels: fast turns caused ghost walls in the odometry-less SLAM map because consecutive scans overlapped too little; `max_vel_theta` was capped at 0.4 rad/s and PWM rotation power reduced to 50%, giving the Ceres Scan Matcher sufficient overlap at every step [10].

## IV. Navigation and Person Detection

### A. Odometry-Less SLAM and Nav2 Navigation

Gmapping was rejected as the SLAM algorithm because its Rao-Blackwellised Particle Filter (RBPF) requires wheel odometry to propagate particle poses between scans. Without encoder data, the filter cannot maintain a stable pose distribution, causing the map to diverge [10], [13].

The SLAM-Toolbox, which extends Karto-SLAM, was selected instead. It builds a pose graph in which robot poses are represented as nodes, and spatial measurements between poses are edges. A Ceres Scan Matcher maximises the overlap between consecutive laser scans to estimate inter-frame motion, so no wheel data is required. When the robot returns to a previously visited location, loop closure detection inserts a new edge into the graph and globally re-optimises all poses, correcting accumulated drift.

Zhang *et al.* [10] confirm that this graph-based, LiDAR-only approach achieves accurate room geometry in structured indoor environments, provided the space contains enough distinctive features such as walls and corners. The SLAM-Toolbox ran in asynchronous mode to further reduce the CPU burden.

Nav2 represents the environment using a three-layer costmap structure. The *static layer* holds walls and fixed structures from the SLAM map. The *obstacle layer* updates in real time with LiDAR detections of dynamic obstacles. The *inflation layer* expands each obstacle by a tuned `inflation_radius`, creating a cost gradient that prevents the global planner from routing the robot too close to walls. The global SMAC Planner (kinematically feasible hybrid A*) computes the optimal route to each navigation goal while respecting the robot's turning constraints. The local controller then generates velocity commands to follow that route while reacting to new obstacles in real time. The global costmap updated at 1.0 Hz to preserve CPU cycles for concurrent person detection. Autonomous area coverage used `explore-lite`, which identifies frontiers between mapped free space and unknown space, selects the nearest reachable frontier, and sends it to Nav2 as the next goal. This continued until no new frontiers remained or a person was detected.

### B. Edge AI Person Detection and Data Fusion

The YOLOv8n model, the smallest variant in the YOLOv8 family, was selected for its low memory demand and high inference speed. Unlike Region-based Convolutional Neural Networks (R-CNN), which evaluate thousands of image regions independently, YOLO processes the full image in a single forward pass, making it suitable for real-time detection on mobile hardware [28]. Running YOLOv8n on the Raspberry Pi 5 CPU caused buffer overflows and ROS 2 communication timeouts because SLAM and Nav2 already consumed most of the available CPU cycles [8]. The solution was to run the model on the Sony IMX500 DSP inside the camera module (Fig. 5) [17]. The Image Signal Processor (ISP) pre-processes raw pixel data on-chip. The DSP then runs the neural network. Only post-inference metadata (bounding boxes, class labels, and confidence scores) is sent to the Raspberry Pi via the CSI interface. Raw frames never reach the host CPU. This design freed the CPU entirely from the inference workload and eliminated all navigation timeouts.

A confidence threshold of 0.5 filtered false positives. Upon a confirmed detection, `ai_camera.py` wrote `person_found = true` to the shared file bridge, causing `explore-lite` to terminate and the robot to stop. OpenCV drew bounding boxes onto the camera image at host level. The annotated JPEG frames were sent to the ROS 2 container via UDP, where the `ros_ai_camera.py` node cross-referenced the bounding-box pixel coordinates with the robot's current pose in the global map frame. This spatial projection saved the victim's estimated map-frame position as a permanent coordinate marker. Combined with the recorded robot trajectory from the start point, this marker gives the rescue team both the victim location and a safe access route without any manual measurement. The Foxglove

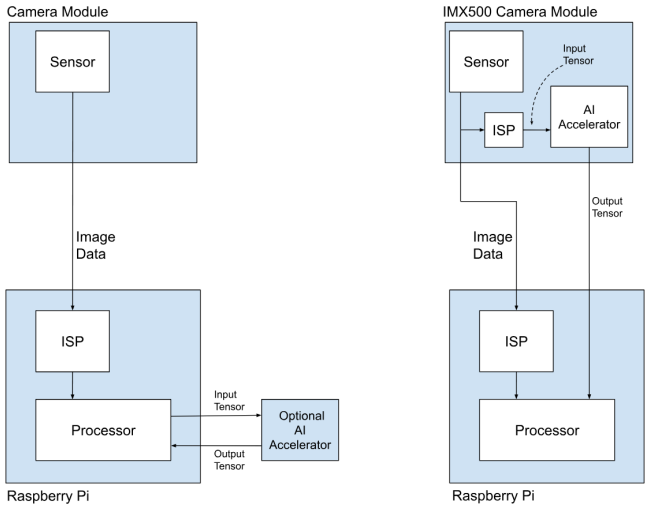



Fig. 5. Inference pipeline comparison. Left: conventional CPU-based approach transfers raw frames to the host for inference. Right: on-sensor processing with the Sony IMX500, only metadata is sent to the Raspberry Pi, freeing the CPU for SLAM and Nav2.

dashboard displayed the live SLAM map with the detection marker, the annotated camera image, a battery voltage gauge (`/battery_state`), and a virtual gamepad teleoperator in one unified view. The Raspberry Pi 5 created a standalone Wi-Fi access point, so no external network infrastructure was required. Fig. 6 shows a completed mission map with all data layers visible.

## V. Experimental Evaluation

### A. Experimental Setup

The system was evaluated in a 55 $m^2$ three-room indoor test course (Fig. 7). Room 1 (3.30×4.50 m) was the fixed robot start area. Room 2 (11.50×2.20 m) was an obstacle corridor containing two repositionable physical barriers (each 1.00×0.25 m) that blocked the direct line of sight and forced the robot to plan detour paths through unknown terrain. Room 3 (3.30×4.50 m) held the simulated victim. All doorways were 1.50 m wide and open. Glass surfaces and reflective structures were removed to prevent LiDAR interference, which had been identified during early development. Access to the test area was restricted during all runs to prevent sensor interference from personnel. The test course floor was flat, dry, and free of loose debris. This study did not test wheel-slip or uneven-terrain conditions, which motivated the odometry-less design, in the physical trials. The first five runs used a fixed obstacle layout. For runs 6–10, the barrier

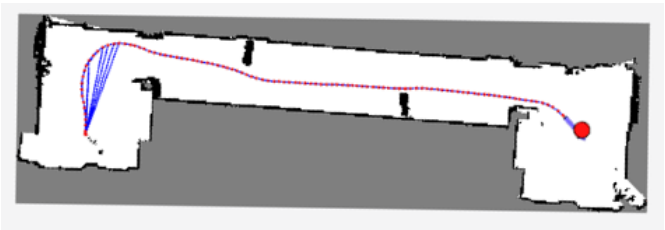


Fig. 6. Final SLAM map after a successful SAR mission: robot trajectory (red), path to last frontier (blue), person detection marker (red dot), free space (white), walls and obstacles (black).

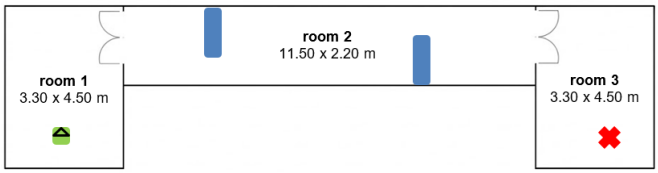



Fig. 7. Test area sketch: Room 1 – start (green square), Room 2 – obstacle corridor (barriers in blue), Room 3 – target area (victim in red).

positions in Room 2 and the victim's location in Room 3 were changed to test robustness under varied conditions. A run was classified as a *success* if the robot autonomously detected the victim within 15 minutes, without a system crash or navigation stall.

### B. System Optimisation

Before the final test series, iterative debugging resolved five critical error classes (Table II). The most impactful change was reducing the global costmap resolution from 5 cm to 10 cm. At 5 cm, the data volume overloaded the ROS 2 message queue during parallel SLAM, navigation, and detection, causing node communication failures. The 10 cm grid reduced costmap memory by a factor of four and eliminated all buffer overflows. Switching from the provisional wooden test chassis to the final 3D-printed chassis also improved stability, because precise component positioning reduced sensor misalignment and vibration-induced errors.

### C. Results and Discussion

Table III summarises all ten test runs. Runs 1–3 failed because the `inflation_radius` was set too large, treating the 1.50 m doorways as impassable. After tuning, runs 4–9 all succeeded without any navigation error. In run 10, the robot completed autonomous navigation and reached Room 3, but the victim had moved to a position outside the camera's fixed field of view. This was not a failure of the YOLOv8n model but a coverage gap in the frontier exploration logic, which stopped before the camera had swept the full room. Detection performance was assessed at the mission level: a run counted as successful once a detection above the 0.5 confidence threshold triggered before the 15-minute limit. Confidence values, false-positive counts, and inference latency were not logged for individual runs, and remain a target for future testing. In all successful runs, the average Time to Detection was 2–3 minutes across the full 55 $m^2$ area. Despite the absence of wheel encoders in every run, all generated SLAM

TABLE II
Critical System Errors and Solutions

| Symptom | Root cause | Solution |
|---|---|---|
| Queue overflow | 5 cm costmap CPU overload | Resolution → 10 cm; update rate reduced |
| TF timeouts | SLAM/Nav2 latency spikes | `transform_tolerance` → 2.0 s |
| Ghost walls | Fast turns; no odometry | `max_vel_theta` → 0.4 rad/s; PWM rotation → 50% |
| Endless spinning | Aggressive Nav2 recovery | Obstacle thresholds raised; `inflation_radius` tuned |
| No frontiers found | Inflation blocking doorways | Inflation layer fine-tuned |

maps correctly reproduced room geometry, wall alignment, and corner angles.

The results confirm three claims. *Odometry-less SLAM is viable on low-cost hardware.* Limiting rotation speed at hardware and software levels gave the Ceres Scan Matcher sufficient scan overlap at every step to produce accurate maps across all ten runs without wheel encoders. This is consistent with Zhang *et al*. [10] and is especially relevant for SAR scenarios where wheel slip on debris makes encoder odometry unreliable [9]. *Edge AI offloading resolves the CPU bottleneck.* After moving YOLOv8n to the IMX500 DSP, all ROS 2 timeouts were eliminated and SLAM ran stably throughout. The CPU remained below saturation during concurrent mapping, navigation, and detection [8], [17]. *The system is cost-competitive.* At £431.68, it costs approximately 30% of a comparable TurtleBot 4 platform and a small fraction of professional SAR systems [6], making fleet deployment and expendable use financially realistic.

The study has six limitations. The 15 mm ground clearance restricts the prototype to flat indoor surfaces, and the trials did not include loose debris or wheel-slip conditions, which motivated the odometry-less SLAM design. Testing under real debris remains future work. Glass and reflective surfaces cause LiDAR interference that degrades the SLAM map. The fixed forward-facing camera field of view produced a coverage gap in run 10, showing that frontier exploration alone does not guarantee full camera sweep of a room. Finally, ten runs is a small sample; validation against NIST standard SAR test methods would be needed for statistically generalisable conclusions. Person detection performance was also assessed only at the mission level; quantitative metrics such as confidence-score distribution, false-positive rate, and inference latency were not recorded and are left for future work.

## VI. Conclusion

This paper presented the design, implementation, and evaluation of a low-cost autonomous SAR robot built on consumer hardware for £431.68. The system combined odometry-less LiDAR SLAM, edge AI person detection, and a unified operator dashboard on a single Raspberry Pi 5.

Testing confirmed the three practical contributions outlined in the introduction. First, LiDAR-only SLAM using the SLAM-Toolbox is a viable alternative to wheel encoder odometry on low-cost indoor platforms. Limiting rotation speed at both hardware and software levels gave the Ceres Scan Matcher sufficient scan overlap to produce accurate room maps without any encoder data, even across ten consecutive runs. Second, offloading YOLOv8n inference to the Sony IMX500 DSP resolved the CPU bottleneck that previously made parallel SLAM and detection impossible on this class of hardware. After offloading, all ROS 2 communication timeouts were eliminated. Third, the Foxglove dashboard consolidated map, camera, battery, and teleoperator data into a single interface accessible on a standard laptop over a local Wi-Fi access point, reducing the cognitive load on operators without specialist training.

TABLE III
Test Run Results Summary

| Run | Result | TTD | Notes |
|---|---|---|---|
| 1–3 | Fail | – | `inflation_radius` too large; path aborted |
| 4–5 | Pass | 2–3 min | First successes after parameter fix; fixed layout |
| 6–9 | Pass | 2–3 min | Varied barriers and victim location; all successful |
| 10 | Fail | – | Navigation complete; detection missed (camera FOV gap) |

The prototype reached a 100% success rate across runs 4–9 after initial parameter tuning, with an average Time to Detection of 2–3 minutes in a 55 $m^2$ three-room course. Across all ten runs, including the three early tuning runs and the later run where the victim was outside the camera's field of view, the combined result was 6 successful runs out of 10. The cost of £431.68 makes the device deployment financially realistic for civil protection organisations.

Five directions are recommended for future work. First, replacing frontier exploration with an Area Coverage Path Planning routine would eliminate camera field-of-view gaps, as observed in run 10. Second, integrating the MPU-6050 gyroscope into an Extended Kalman Filter would combine inertial data with LiDAR odometry, improving rotation accuracy during fast manoeuvres [11]. Third, the Raspberry Pi 5 Neural Processing Unit expansion module could enable additional AI capabilities such as vital sign detection. Fourth, increasing chassis ground clearance, adding weatherproofing, and testing on loose debris and uneven terrain would extend the system beyond flat indoor surfaces to real disaster environments. Fifth, future test runs should log quantitative detection metrics, such as confidence-score distributions, false-positive rates, and inference latency, to support a more detailed evaluation of person-detection performance.

## References


[1] A. G. Macintyre, J. A. Barbera, and E. R. Smith, "Surviving collapsed structure entrapment after earthquakes: A 'Time-to-Rescue' analysis," *Prehospital and Disaster Medicine*, vol. 21, no. 1, pp. 4–17, Feb. 2006, doi: 10.1017/s1049023x00003253

[2] R. Voyles and H. Choset, "Editorial: Search and rescue robots," *Journal of Field Robotics*, vol. 25, no. 1–2, pp. 1–2, Jan. 2008, doi: 10.1002/rob.20231

[3] J. Casper and R. R. Murphy, "Human-robot interactions during the robot-assisted urban search and rescue response at the World Trade Center," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 33, no. 3, pp. 367–385, Jun. 2003, doi: 10.1109/TSMCB.2003.811794

[4] R. R. Murphy, "Human–robot interaction in rescue robotics," *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, vol. 34, no. 2, pp. 138–153, May 2004, doi: 10.1109/TSMCC.2004.826267

[5] G. D. Cubber *et al*., *Search and Rescue Robotics - From Theory to Practice*. InTech, 2017.

[6] J. Delmerico *et al*., "The current state and future outlook of rescue robotics," *Journal of Field Robotics*, vol. 36, no. 7, pp. 1171–1191, Oct. 2019, doi: 10.1002/rob.21887

[7] H. Surmann *et al*., "Lessons from robot-assisted disaster response deployments by the German Rescue Robotics Center task force," *Journal of Field Robotics*, vol. 41, no. 3, pp. 782–797, May 2024, doi: 10.1002/rob.22275

[8] R. Vaghela *et al*., "Optimizing object detection for autonomous robots: A comparative analysis of YOLO models," *Measurement*, vol. 257, Aug. 2025, Art. no. 118676, doi: 10.1016/j.measurement.2025.118676

[9] K. Nagatani, D. Endo, and K. Yoshida, "Improvement of the odometry accuracy of a crawler vehicle with consideration of slippage," in *Proc. IEEE Int. Conf. on Robotics and Automation*, Rome, Italy, 2007, pp. 2752–2757, doi: 10.1109/ROBOT.2007.363881

[10] X. Zhang, L. Guokun, F. Genping, X. Dongliang, and L. Shiliu, “SLAM algorithm analysis of mobile robot based on lidar,” in *Proc. Chinese Control Conference (CCC)*, Guangzhou, China, 2019, pp. 4739–4745, doi: 10.23919/ChiCC.2019.8866200

[11] R. R. Murphy, *Introduction to AI robotics*, 2nd ed. Cambridge: MIT Press, 2019.

[12] L. Nwankwo, C. Fritze, K. Bartsch, and E. Rueckert, “ROMR: A ROS-based open-source mobile robot,” *HardwareX*, vol. 14, Jun. 2023, Art. no. e00426, doi: 10.1016/j.ohx.2023.e00426

[13] Z. Yang, N. Naz, P. Liu, and M. N. Huda, “Evaluation of SLAM Algorithms for Search and Rescue Applications,” in *Proc. Towards Autonomous Robotic Systems (TAROS 2023)*, Cambridge, UK, 2023, pp. 114–125, doi: 10.1007/978-3-031-43360-3_10

[14] N. Tahir and R. Parasuraman, “Edge computing and its application in robotics: A survey,” *Journal of Sensor and Actuator Networks*, vol. 14, no. 4, Jun. 2025, Art. no. 65, doi: 10.3390/jsan14040065

[15] M. Beltran-Escobar, T. E. Alarcon, J. Y. Rumbo-Morales, S. Lopez, G. Ortiz-Torres, and F. D. J. Sorcia-Vazquez, “A review on resource-constrained embedded vision systems-based tiny machine learning for robotic applications,” *Algorithms*, vol. 17, no. 11, Oct. 2024, Art. no. 476, doi: 10.3390/a17110476

[16] C. Kong, F. Li, X. Yan, J. Yang, P. Mo, Q. Luo, and R. Mao, “Object detection on low-compute edge SoCs: A reproducible benchmark and deployment guidelines,” *Scientific Reports*, vol. 16, no. 1, Jan. 2026, Art. no. 5875, doi: 10.1038/s41598-026-36862-y

[17] Raspberry Pi AI Camera, Raspberry Pi Documentation. [Online]. Available: https://www.raspberrypi.com/documentation/accessories/ai-camera.html

[18] Raspberry Pi Documentation. [Online]. Available: https://www.raspberrypi.com/documentation/computers/raspberry-pi-5.html

[19] Raspberry Pi Datasheets. [Online]. Available: https://datasheets.raspberrypi.com/rpi5/raspberry-pi-5-product-brief.pdf

[20] Shanghai Slamtec Co., RPLIDAR C1 360° Laser Range Scanner Datasheet, Slamtec Datasheets. [Online]. Available: https://download-en.slamtec.com/api/download/rplidar-c1-datasheet/1.2?lang=en

[21] Raspberry Pi AI Camera Product Brief, Raspberry Pi Documentation. [Online]. Available: https://datasheets.raspberrypi.com/camera/ai-camera-product-brief.pdf

[22] TT Motor All-Metal Gearbox 1:90 Gear Ratio Datasheet, Adafruit Product Files. Available: https://cdn-shop.adafruit.com/productfiles/3802/C11454_datasheet_TT1___90_metal.pdf

[23] Waveshare, UPS Module 3S Uninterruptible Power Supply Datasheet. [Online]. Available: https://www.waveshare.com/wiki/UPS_Module_3S

[24] US-100 Ultrasonic Distance Sensor Datasheet. [Online]. Available: https://media.digikey.com/pdf/Data%20Sheets/Adafruit%20PDFs/4019_Web.pdf

[25] NVIDIA Corporation, NVIDIA Jetson Nano Developer Kit Data Sheet. [Online]. Available: https://cdn.sparkfun.com/assets/0/7/f/9/d/jetson-nano-devkitdatasheet- updates-us-v3.pdf

[26] InvenSense Inc., MPU-6000 & MPU-6050 Product Specification. [Online]. Available: https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000- Datasheet1.pdf

[27] 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, May 2022, Art. no. eabm6074, doi: 10.1126/scirobotics.abm6074

[28] S. Dinesh Babu, A. Saranyadurai, M. Noor Ali, N. Arunfred, N. Ganesh, and S. Sujatha, “Autonomous robots for disaster relief with IoT and YOLOv3 object detection,” in *Proc. 6th Int. Conf. on Inventive Research in Computing Applications (ICIRCA)*, Coimbatore, India, 2025, pp. 687–692, doi: 10.1109/ICIRCA65293.2025.11089612