International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 11 | Nov - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Map-Aware Autonomous Indoor Delivery Robot with Built-In Self-Recovery for Smart Campus Applications

Abhay B S[1]*, Khushal Tak C[2], Abhishek S Jambagi[3], Darshan Raju Paiginikar[4], Dr. Darshan B D[5]

[1 2 3 4] Student, Department of Electronics and Communication Engineering, SJB Institute of Technology, Bengaluru, India
[5] Assistant Professor, Department of Electronics and Communication Engineering, SJB Institute of Technology, Bengaluru, India

[1]abhaybs202@gmail.com, [2]khushalkt7@gmail.com, [3]abhishek.jambagi2005@gmail.com, [4]darshanpai394@gmail.com, [5]darshanbd@sjbit.edu.in (*corresponding author)

Abstract—Files, books and lab components are carried by hand between rooms in colleges and offices, and low-cost delivery robots usually follow a fixed line or wait for a person when something goes wrong. We built a low-cost indoor delivery robot on ROS 2 Foxy and Nav2, with a Raspberry Pi 4B, an Arduino Uno with encoder motors, a YDLIDAR X2 and a servo-closed parcel lid; users request and track deliveries in a Firebase web app. A self-recovery layer runs inside every delivery. It detects a lost position from a scan-to-map match score and finds the robot again with a whole-map search. It waits, goes round or detours when the route is blocked. It lines the front-wheel robot up with narrow gaps. It restarts a faulty motor controller. After each recovery the same leg continues. In 32 evaluable real deliveries in two houses and at a national project expo, the robot completed 25 (78 %, 95 % CI 61–89 %). Of 49 resolved position losses, 41 were recovered without help (median 4.0 s), and 70 of 80 narrow-gap manoeuvres ended without help. In 18 later jobs in a third house, all 16 jobs not cancelled or stopped on purpose were delivered, the robot found its position after 7 deliberate kidnaps (5 of them by itself within 4 s), and the battery prediction declined or safely ended jobs in all 3 low-battery trials. Its parts cost about Rs. 28,200.

Keywords—Indoor delivery robot, ROS 2, Nav2, LiDAR localisation, self-recovery, IoT.

I. INTRODUCTION

Autonomous mobile robots now carry goods in warehouses, hospitals and hotels, mainly because small 2D LiDAR sensors became cheap and good navigation software became free: the Robot Operating System 2 (ROS 2) [1] and its navigation stack Nav2 [2]. With these tools, a robot can build a map of a building, find its own position on that map and plan a safe path to any point in it.

Inside a typical Indian college, however, files, letters, books and lab components are still carried by hand between the office, the staff rooms, the library and the labs. Each trip is short, but there are many of them in a day. Commercial delivery robots could do this work, but they are far too costly for most colleges. Many student projects build line-following robots, which are cheap but stop as soon as a chair is placed on the line and cannot reach any room that has no line.

A map-based robot avoids these problems. Our first tests showed that reaching a goal in an empty room is easy, but finishing deliveries in a real house is not. The robot lost its position after turning on a smooth floor. It could not pass a 70 cm doorway, because its drive wheels are at the front and its back swings out. It stopped whenever a person stood in the corridor, and its motor controller sometimes reset when the battery voltage dipped. Every one of these failures needed a person to walk over and help. A delivery robot that has to be rescued is of little use, so this work makes recovery from such failures a normal part of every delivery.

The objectives of the work were to: (i) build a low-cost differential-drive robot with ROS 2 control; (ii) map an area by hand or by autonomous exploration and navigate between named delivery spots; (iii) provide a web app with a job queue and a parcel lid that the robot opens only for the sender and the receiver; (iv) recover on its own from lost position, blocked routes, narrow passages and motor faults and then continue the same delivery; and (v) measure how well all of this works over many real deliveries. Section II reviews related work, Section III describes the system and the recovery methods, Section IV presents the results, Section V discusses them, and Section VI concludes the paper.

II. LITERATURE REVIEW

A. Navigation Building Blocks

ROS 2 [1] provides the communication layer on which our robot is built, and Nav2 [2] provides localisation with AMCL, path planning, path following and generic recovery actions such as clearing costmaps, spinning and waiting, all controlled by a behaviour tree. The Nav2 authors ran robots for a 37-mile "marathon" on a university campus and treated recoveries as a normal part of a fault-tolerant system rather than as failures. SLAM Toolbox [3] builds the 2D maps, and frontier-based exploration [4] is still the standard way to map a building automatically. Our work uses these components and adds a layer above them at the level of the delivery job.

B. Low-Cost Indoor Delivery Robots

Liu et al. [5] built a low-cost ROS 2 package vehicle on Nav2 and Cartographer with LiDAR, camera and IMU fusion. Kim et al. [6] developed a multi-floor delivery robot that uses lifts and tested it for a month in a real building, on a much costlier platform. Ibrahim et al. [7] accept item requests through a web portal on a Jetson Nano robot and confirm the hand-over with QR codes, and Cui et al. [8] use an Arduino, a servo-driven storage box and SMS alerts for file transfer. Mhadeshwar et al. [9] guide a Nav2 TurtleBot 4 to QR-coded hospital destinations, while Shrestha et al. [10] and Patil et al. [11] use Arduino- and Raspberry Pi-class hardware close to ours.

These works show that low-cost robots can navigate and deliver, but they rarely describe what happens when localisation, the route or the motor controller fails during a delivery, and few report success rates. An exception is the college document-delivery robot of Akinwumi et al. [12], which completed 60 % of its test routes without human help.

C. Field Trials and Recovery

The STRANDS project [13] reported long-term autonomy of service robots in offices and care homes, measured by system lifetime and the share of time spent autonomous. Valner et al. [14] ran a robot fleet carrying samples in a hospital on ROS 2, and Lee et al. [15] analysed 122 medication-delivery missions and found that most failures came from people or objects blocking the robot. These studies use research-grade or commercial platforms. Finding a lost ("kidnapped") robot with a 2D LiDAR is well studied. Ismail and Bukhori [16] detect kidnapping with a dynamic-time-warping test, and Akai [17] estimates how reliable the pose is within Monte Carlo localisation. Zhang et al. [18] recover the pose from one scan by scoring hypotheses in stages, and Du et al. [19] start a branch-and-bound scan-matching search when localisation fails. Our position recovery follows the same idea in a simpler form. For other faults, De Luca et al. [20] organise recovery behaviours in a behaviour tree and MROS [21] adapts ROS control architectures at run time.

D. Research Gap

To our knowledge, the low-cost delivery robots reviewed above do not handle lost position, blocked routes, narrow gaps and motor-controller faults together inside the delivery job, and they seldom report how often such recoveries happen in real deliveries. Table I summarises this comparison. The contributions of this paper are therefore: (i) a self-recovery layer that runs inside the delivery job of a Nav2 robot and resumes the same leg after each recovery; (ii) a working low-cost delivery system with a web app, a servo-closed parcel lid, battery monitoring and autonomous mapping; and (iii) an evaluation on the real robot over 40 logged delivery jobs, including a live public demonstration, and 18 further jobs that test repeated deliveries, kidnapping and low battery.

TABLE I. COMPARISON OF RELATED DELIVERY ROBOTS WITH THE PROPOSED SYSTEM

System	Navigation	User interface	Recovery reported
Ibrahim et al. [7]	SLAM + AMCL, Jetson Nano	Web portal, QR	Not described
Kim et al. [6]	3D maps, lift use	Task planner	Not described
Cui et al. [8]	ROS, particle filter	Qt app, SMS	Not described
Mhadeshwar et al. [9]	Nav2 + QR codes	Not described	Not described
Akinwumi et al. [12]	ESP32, ultrasonic, GPS	Not described	Not described
Proposed	**Nav2, AMCL, own explorer**	**Firebase web app**	**Position, route, gap, motor**

III. METHODOLOGY

A. Hardware Platform

The robot, shown in Fig. 1, is a closed box about 33 cm long, 24 cm wide and 19 cm tall. Its two drive wheels (70 mm diameter, 0.288 m apart) are at the front of the body and a swivel caster supports the rear, which leaves the middle of the body free for the parcel compartment. The YDLIDAR X2 on top gives a 360-degree scan of the surroundings about 7 times per second (the datasheet value at its recommended motor speed), with a range of up to 8 m. A Raspberry Pi 4B (4 GB) runs the LiDAR driver and the programs that talk to the hardware. An Arduino Uno with the ROSArduinoBridge firmware runs a PID speed loop for each wheel from the 1024-count encoders and drives the motors through an L298N. An Arduino Nano moves an SG90 servo that opens the lid to 70 degrees and closes it to 0 degrees. An HC-SR04 ultrasonic sensor watches the floor area in front of the robot, below the LiDAR's scan plane, and an INA219 sensor on the I2C bus measures the voltage and current of the motor battery: three 2500 mAh 18650 lithium-ion cells in series (11.1 V nominal, 12.6 V full, about 27.75 Wh, of which about 26 Wh lies above the 9.5 V cut-off used in the software). The Raspberry Pi has its own power bank, so a voltage dip in the motor battery cannot reset it. Table II lists the main parts and their cost.

TABLE II. MAIN HARDWARE COMPONENTS AND COST

Component	Role	Cost (Rs.)
Raspberry Pi 4B, 4 GB	Robot-side computer	9,600
YDLIDAR X2	Mapping and localisation	6,547
2 × encoder DC gear motors	Drive and odometry	1,428
Arduino Uno, Arduino Nano, L298N	Motor and lid control	800
HC-SR04, INA219, servo, modules	Sensing and lid	3,388
Li-ion cells, charger, power bank	Power	2,350
Chassis, caster, wiring, SD card, misc.	Structure	4,111
Total (parts)		**28,224**



Fig. 1. The delivery robot (LiDAR at the rear of the top, parcel lid at the front)

B. Software Architecture The software, shown in Fig. 2, is split across four layers. Users work in a web app hosted on Firebase, which stores requests, the robot's state, the delivery spots and an event timeline in a Realtime Database protected by sign-in and security rules. A laptop running Ubuntu 20.04 and ROS 2 Foxy runs the navigation and the job logic: AMCL, Nav2, the delivery manager, the self-recovery nodes and a bridge node that reads new requests and button presses from Firebase every 1.5 s and writes the robot's status back. The Raspberry Pi on the robot runs the LiDAR driver, ros2_control with the differential-drive controller, a safety watchdog, the ultrasonic guard, the battery monitor and the lid controller. The laptop and the Pi communicate over Wi-Fi through the standard ROS 2 middleware.

Several nodes may want to move the robot at the same time, so all speed commands go through twist_mux, which passes on only the highest-priority active input. Nav2 has the lowest priority (10), followed by the path guard (40), the gap assist (45), the ultrasonic guard (47), the position recovery (50) and finally the keyboard and the emergency stop (100). When a safety node stops publishing, control returns to Nav2 automatically within 0.3 to 0.5 s.

C. Delivery Workflow

The delivery manager runs each job as the state machine in Fig. 3. A sender chooses a receiver in the web app, and the job enters a queue in which urgent jobs go first, then any job that has waited more than 300 s, and otherwise the job whose sender is nearest to the robot. Before every drive the lid is closed, and the manager waits until the lid controller reports that the servo has reached the closed angle (there is no lid sensor). At the sender's spot the robot waits for the sender to press "Open", load the parcel and press "Close"; the security rules allow only a member of the sender's cabin to press these buttons, and only while the robot waits there. The robot then drives to the receiver, where only the receiver can open the lid. If the receiver's spot cannot be reached, the robot brings the parcel back to the sender instead of leaving it somewhere on the way. After each job, the robot either takes the next job directly or returns home.

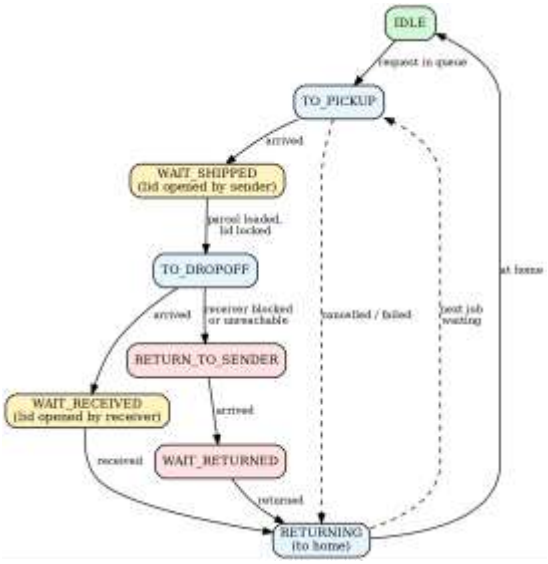

Fig. 3. State machine of the delivery manager

The delivery spots are set by an administrator on the web app, either by tapping on the map or by driving the robot to the spot. A spot is accepted only if it lies on free floor at least 0.25 m from walls and can be reached from Home by a robot of 0.17 m radius. A goal counts as reached when Nav2 reports success and the robot is within 0.5 m of the spot, or when Nav2 gives up (or only keeps turning for 8 s) within 0.35 m of it.

D. Self-Recovery Layer

This section describes the recovery behaviours, which run during every delivery. Each one pauses the current leg; when it succeeds, the delivery manager sends the same goal again, so the user sees a short delay instead of a failed delivery.

1) Lost Position

AMCL gives no clear signal when its estimate is wrong, so a

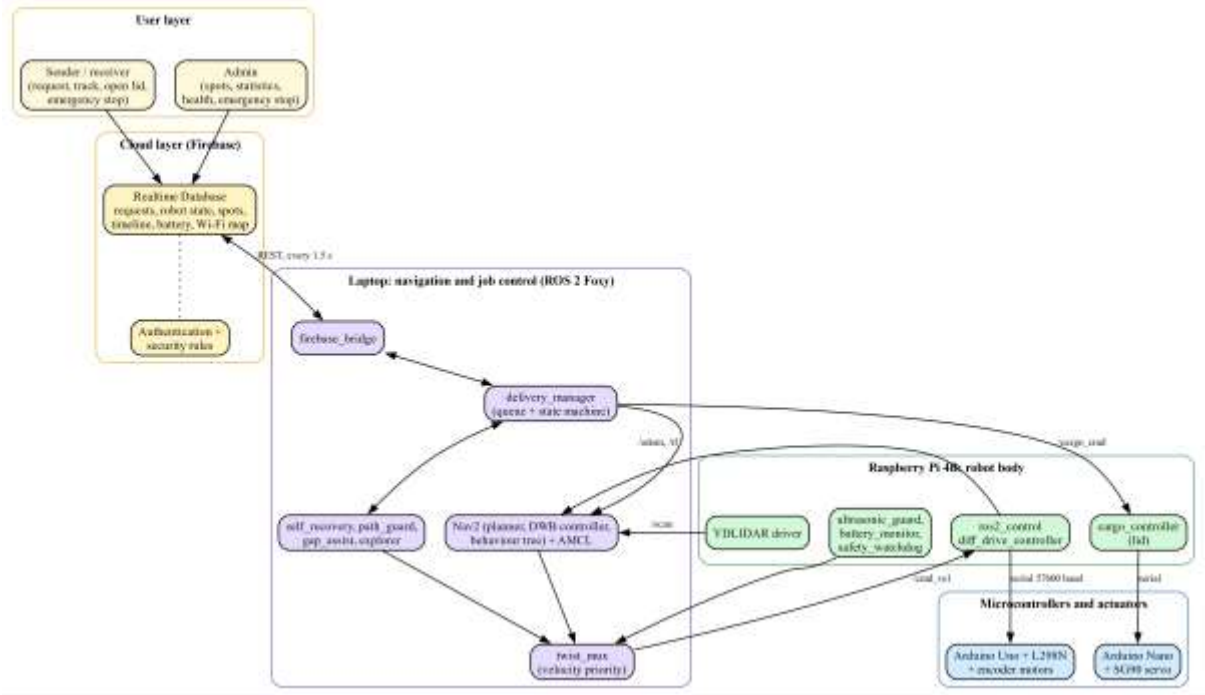

Fig. 2. Software architecture: web app, Firebase, laptop (navigation and job control) and robot

separate node checks every scan against the map twice a second. Each scan point is placed on the map at the current pose, ignoring points closer than 0.2 m (the robot's own body or the open lid) and farther than 6 m. The match score m is the share of scan points that lie within two map cells (10 cm) of an occupied cell:

$$m = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}[d(p_i) \le 0.10\ \text{m}] \qquad (1)$$

where N is the number of scan points used and $d(p_i)$ is the distance from the i-th scan point p_i to the nearest wall in the map. In Section IV, m is given as a percentage (0.81 = 81 %). A well-localised robot normally scores 0.8 or more. When m stays below 0.50 for 2 s, the robot is declared lost: the leg is paused, the robot stops, and a global search is run (Fig. 4). The search takes up to 90 scan points and places them at every free map cell at least 15 cm from the walls, on a 0.2 m grid, for 24 headings at once, using vectorised array operations. It keeps the 8 best cells per heading. The 10 best candidates are then refined on a finer grid (±0.2 m in 5 cm steps, ±10 degrees in 2.5 degree steps) with all scan points and scored with a tighter 5 cm tolerance; this refined score is the one compared with 0.75. The best pose is accepted only if it scores at least 0.75 and beats the best candidate that is more than 0.5 m or 30 degrees away by at least 0.05, which protects against look-alike places. The pose is then given to AMCL and verified over three good checks. If no place is clearly best, the robot turns 60 degrees for a new view, but only if no LiDAR point is within 0.33 m of its axle. It searches again, up to six times, and then asks for help on the website. While it waits for help it keeps searching every 5 s, so a robot that has been moved into open space recovers by itself. Because the scan lags behind the pose during fast turns, a low score within 1.5 s of a turn faster than 0.5 rad/s is ignored. One search took 0.1 to 0.3 s on the laptop (0.2 s in most of the 22 logged searches). The same search runs once at start-up, because a robot placed even a little off its Home spot started with a poor match that AMCL alone did not correct. At start-up the new pose needs 0.70 instead of 0.75 and must beat the given pose by 0.10, because Home is only a guess. AMCL's own random-particle recovery is switched off so that only this verified search moves the pose.

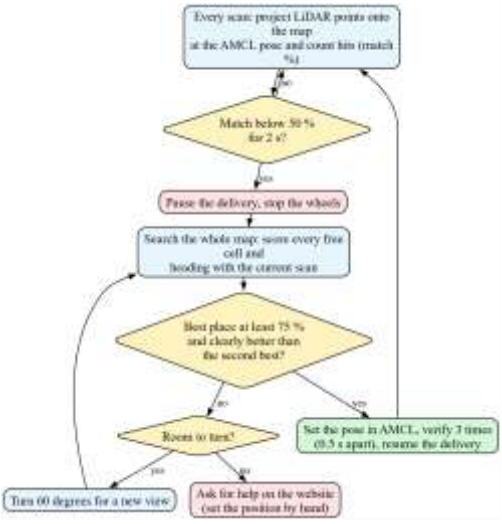

Fig. 4. Flowchart of the lost-position recovery

2) Blocked Routes

A path guard checks the next 1.5 m of the planned route for LiDAR points that are not part of the map walls. If at least three such points lie within 0.22 m of the route and closer than 0.8 m, it first lets Nav2 produce one new plan, and stops the robot only if that plan still runs into the obstacle (or at once if it is closer than 0.5 m). It then waits up to 4 s, because people usually step aside; if Nav2 finds a way round on the same route, the robot continues. Otherwise the blocked points are remembered for up to 180 s in the global costmap, so that Nav2 plans a different route (a detour). If no route exists, the robot reports the blocked route on the website and returns the parcel or goes home.

3) Narrow Gaps

Because the wheels are at the front, the rear corners of the robot sweep a circle of about 0.27 m radius round the axle centre when it turns on the spot, and Nav2's controller often finds "no valid trajectories" in front of a doorway. A gap-assist node detects this condition (the controller's message plus less than 3 cm of movement in 2 s) and plans a short sequence of manoeuvres with a best-first search over turns of 15 or 45 degrees about the axle centre or about either wheel and forward moves of 0.10 or 0.30 m. Every move is checked against the live LiDAR points for the robot's whole swept outline. When the robot faces along the route with at least 0.40 m free ahead, it drives straight for up to 0.45 m through the gap. If that straight drive is blocked, it backs up by 0.12 to 0.35 m, turning its nose away from the closer side, but only while at least 0.10 m is free behind. After each manoeuvre it asks AMCL for three extra updates and hands control back to Nav2, with at most four assists per goal.

4) Motor Faults

A watchdog on the Raspberry Pi stops the motors in four cases: a wheel reading jumps faster than 30 rad/s (the Arduino has reset); drive commands are active for 5 s but the wheels do not turn; a wheel races while reversing; or both wheels turn more than 2 rad/s the wrong way for 1 s (a fault found in November). It then stops the motor controller, sends stop commands to the Arduino directly over its serial port and restarts the controller. It reports "RESTARTED" once wheel readings return, or "GAVE UP" if no wheel readings arrive within 45 s or after 3 restarts in 10 minutes. The delivery manager pauses the leg during this time and continues it afterwards. Table III lists the main parameters of the recovery layer.

TABLE III. MAIN PARAMETERS OF THE SELF-RECOVERY LAYER

Parameter	Value
Lost threshold / duration	m < 0.50 for 2 s
Search grid / headings	0.2 m / 24 (15 degrees)
Accept / uniqueness margin	m ≥ 0.75 / 0.05
New-view turn / max searches	60 degrees / 6
Path guard look-ahead / wait	1.5 m / 4 s
Blocked-point memory	180 s
Gap assist straight drive	up to 0.45 m
Back-up distance	0.12 to 0.35 m
Motor fault: jump / frozen	> 30 rad/s / 5 s
Motor fault: wrong way	> 2 rad/s for 1 s
Restart timeout / limit	45 s / 3 in 10 min

E. Mapping, Safety and Energy

A new place can be mapped in two ways. The robot can be driven by hand while SLAM Toolbox builds the map, or an explorer node can map it on its own. The explorer repeatedly drives to the best frontier [4], scored by its length minus the driving distance and the turning needed. It blacklists frontiers

that cannot be reached and finally returns to its start point, which becomes Home.

The ultrasonic guard stops all forward motion and turning when a real echo shows an object closer than 0.20 m. It keeps the stop for 3 s after the last close echo and releases it on a real echo beyond 0.25 m. When Nav2 keeps pushing forward, the guard backs the robot off, and it marks the object in Nav2's local costmap. In addition, the speed is limited to 0.24 m/s and 0.5 rad/s, the Arduino firmware stops the motors if no command arrives for 2 s, a rocker switch on the robot and an emergency stop on the web app cut the drive, and the robot never drives while the lid may be open. The robot has no bumper, so a team member stayed near it during all tests.

The battery monitor reads the INA219 once a second. It corrects the voltage for the pack's internal resistance and converts the resting cell voltage into a usable charge percentage, with 0 % at 9.5 V, below which the motor driver no longer works properly. The delivery manager uses this percentage to predict whether the rest of a job can be finished with 12 % to spare. The prediction uses the straight-line distance through the remaining stops, an energy-per-metre figure (a cautious first guess of 0.05 Wh per metre, which the robot then learns from its own legs) and 0.25 Wh for each stop. A request that cannot be completed is declined with an apology. During a job, a shortfall that lasts 10 s makes the robot stop safely. It goes home if nothing is loaded and hands the parcel back if it is still at the sender. With the parcel on board, it returns the parcel if the sender is nearer, otherwise finishes the delivery if the energy fits, and otherwise goes home with it. This prediction was added after the October tests and was checked in scripted tests and in three real trials in November.

F. Experimental Setup

The robot was tested on four days in October 2026 and one day in November 2026. House 1 (several rooms, marble floor, map built by the explorer) was used on two days, House 2 (a 22 m² area, map built by the explorer) on one day, and the National Level Project Expo 2026 at SJBIT, a crowded hall mapped by hand, on one day. On 12 November, in House 3 (a home mapped by hand, with three cabins 2.0 to 3.4 m apart), the team ran repeated deliveries between the same cabins without the team cancelling any job. A team member also lifted the robot during deliveries and put it down elsewhere (kidnap tests), and the battery prediction was tested on the real, nearly empty battery by changing the reserve during a job. Every delivery was requested from a phone or laptop through the web app. A team member stayed near the robot with the motor switch as an emergency stop, and touched the robot only when the website asked for help or to stop it with the emergency stop. Every event was written to a log file with its time and the robot's pose, and all the results below were computed from the event log and the node logs (Wi-Fi map, explorer). Jobs that the team cancelled during testing are reported separately and are not counted as successes or failures. Success rates are given with Wilson 95 % confidence intervals, which suit small samples. Times are summarised by the median and the 90th percentile, because a few long waits make their distributions skewed. The relation between leg time and path length is described by a least-squares line and Pearson's correlation coefficient r.

IV. RESULTS

A. Mapping

The explorer mapped House 2 (22.2 m²) in 4.2 min, driving 28.3 m, and returned to its starting point without help. Fig. 5 shows this map with the delivery spots that were later set on the web app. In the expo hall, the walls were far away and often hidden by people, so only the stall and the floor in front of it were mapped by hand.

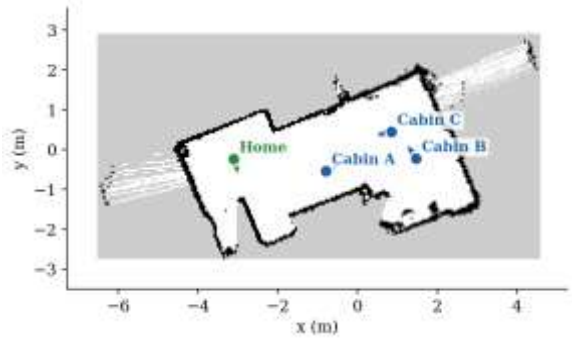

Fig. 5. Map of House 2 built by autonomous exploration, with Home and three cabins

B. Delivery Outcomes

Forty delivery jobs were logged. The robot completed 25 deliveries, brought 2 parcels back to the sender, failed 5 jobs, and 8 jobs were cancelled by the team. Excluding the cancelled jobs, the success rate was 25 of 32, or 78 % (Wilson 95 % confidence interval 61–89 %). Five further jobs were started but have no logged outcome because the delivery manager was restarted during them; counting all of them as failures would give 25 of 37 (68 %). Table IV and Fig. 6 break the results down by session.

TABLE IV. DELIVERY OUTCOMES PER SESSION

Session	Delivered	Returned	Failed	Success
House 1, 27 Oct	5	0	0	5/5
House 1, 28 Oct	1	0	2	1/3
House 2, 29 Oct	8	1	0	8/9
Expo, morning	0	0	1	0/1
Expo, afternoon	11	1	2	11/14
Total	**25**	**2**	**5**	**25/32 (78 %)**

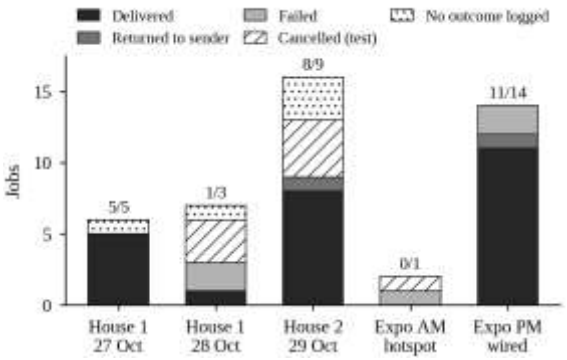

Fig. 6. Job outcomes per session (cancelled jobs shown separately)

The worst session was 28 October: one job failed after two motor faults, and in the other the robot could not get close enough to a spot in a tight corner of House 1. The gap assist had been changed on the evening of 27 October to keep one turning direction and not to act near the goal; in House 2 on 29 October the robot completed 8 of 9 jobs. At the expo, the morning produced no delivery. The robot used a phone

hotspot in a crowded hall and a battery pack ran flat; after the pack was replaced with a charged one, two motor-controller restarts still failed. In the afternoon, with a router near the stall and a new map, it completed 11 of 14 jobs.

C. Travel Time

The robot drove 257.8 m over 112 legs, of which 94 reached their goal. The median leg took 19.7 s, or 10.5 s per planned metre. Fig. 7 shows that leg time grows steadily with path length when nothing happens on the way (5.2 s/m, r = 0.79, 53 legs), but is spread widely when a wait, a narrow gap or a recovery occurs (r = 0.15, 41 legs). In 41 of the 94 completed legs at least one such event happened, and in 40 of them the robot finished the leg by itself (in 1 a person helped). A person had to help on 4 of the 112 legs. Separately, the team pressed the emergency stop 16 times in October and 6 times on 12 November.

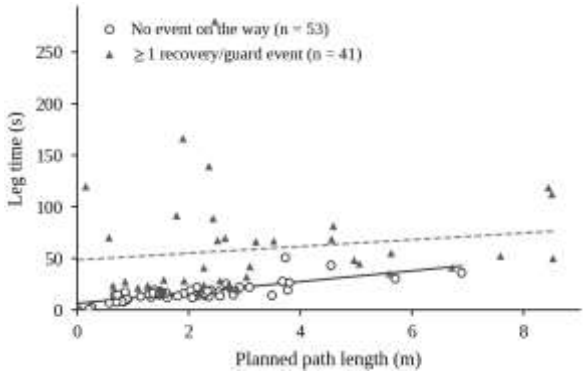

Fig. 7. Leg time against planned path length, with and without events on the way

D. Self-Recovery Performance

The robot was declared lost 60 times. In 41 cases it found its position again without help, with a median of 4.0 s, a 90th percentile of 12.5 s and a maximum of 16.5 s; 36 of these took less than 5 s, and 5 needed a turn for a new view (Fig. 8). In 8 cases a person had to help (move the robot or set the pose). 11 episodes have no logged end: 10 because the software was restarted and 1 because the log ended; in 10 of these 11 the robot had already asked for help. At start-up, in 8 restarts where the robot was placed off its Home spot, the starting match was between 6 and 61 %, and the global search raised it to between 72 and 93 % (median 81 %).

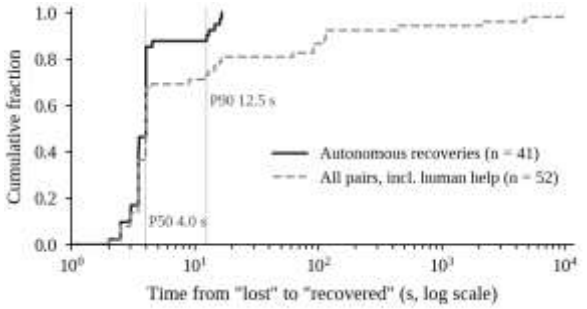

Fig. 8. Cumulative distribution of autonomous recovery time

The gap assist detected a stuck robot 78 times and performed 80 manoeuvres (Fig. 9). In 50 (62.5 %, 95 % CI 52–72 %) the robot drove through the gap and Nav2 continued, and in 20 (25 %) the navigation goal ended during the manoeuvre, so 70 of 80 (87.5 %) ended without help. The path guard stopped the robot in 38 separate waiting episodes, let it go round obstacles 32 times, took 3 detours and reported a blocked route 12 times. The watchdog detected 7 motor faults: in 4 the controller restarted by itself within 20 to 25 s and the delivery continued, and in 3 it gave up and asked for help. Table V summarises these numbers.

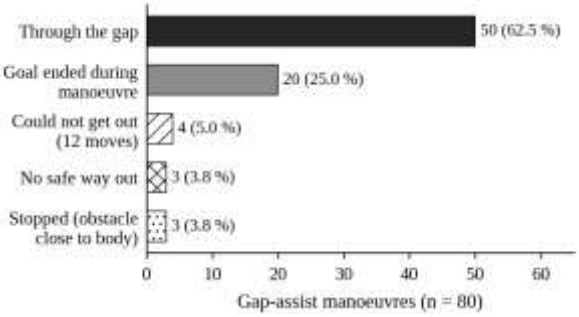

Fig. 9. Outcomes of the 80 gap-assist manoeuvres

TABLE V. SUMMARY OF RECOVERY EVENTS IN THE REAL TESTS

Behaviour	Result
Lost position (60 episodes)	41 autonomous, median 4.0 s
Start-up check (8 restarts)	6–61 % raised to 72–93 %
Gap assist (80 manoeuvres)	70 ended without help (50 through)
Path guard	38 waits, 32 go-rounds, 3 detours
Motor faults (7)	4 restarted, 3 gave up
Legs needing a person	4 of 112

E. Repeat, Kidnap and Battery Tests in House 3

Table VI summarises the November session. Of 18 jobs, 16 were delivered; one was cancelled by the sender and one was stopped on purpose by the battery trial, so no job failed. Two motor faults occurred late in the session, when the battery was at about 28 % and 20 %. In the first, the Arduino reset and the watchdog restarted the controller in 22 s. Afterwards the Arduino's speed loop drove both wheels backwards at full speed for any forward command, which the robot reported as repeated position losses. On this return leg the fault needed people: the emergency stop was pressed several times, and resetting the Arduino by hand (opening its serial port) and restarting the motor controller cured it. Later, the new wrong-way rule caught a second occurrence by itself and restarted the controller in 21 s, and the robot continued its trip home. Most short position losses in this session came while the motor fault made the robot drive backwards, or right after the Arduino reset; one came during a normal turn. In each case the search found almost the same place again (98–100 % match). A rule that ignores the match for 1.5 s after a fast turn (Section III-D) was added during the session but was not tested on its own.

TABLE VI. RESULTS OF THE NOVEMBER TESTS IN HOUSE 3

Test	Result
Repeated deliveries (14 jobs)	12 delivered, 1 cancelled, 1 battery stop
Kidnap tests (4 jobs, 7 kidnaps)	4 delivered, all 7 kidnaps recovered (5 without help)
Kidnap, put down in open space	5 of 5 without help, median 4.0 s
Kidnap, put down near a wall	2 of 2 after one move, 19.5 s and 38 s
Battery: request at 9 %	Declined at once with an apology
Battery: shortfall at the sender	Job stopped before pick-up, went home
Battery: shortfall with the parcel	Finished the delivery, then went home
Motor faults (2)	Both restarted automatically

In the kidnap tests, a team member lifted the robot while it was driving and put it down 1.2 to 2.7 m from where it had been, turned by up to 90 degrees. When it was put down in open space, it found its position in 3.5 to 4.0 s every time and continued the same leg. When it was put down near a wall, it had no room to turn and asked for help; once it was moved, the repeated search found it without anyone setting the pose. In one of these cases, the first candidate (81 %) did not pass the verification, and the robot turned for a new view and accepted the correct pose (99 %). In the battery trials, the reserve was raised from 5 % to 12 % during a job on the real battery: before pick-up the robot stopped the job and went home with an apology, and with the parcel on board, close to the receiver, it chose to finish the delivery before going home to charge. A request made at 9 % was declined within 1.2 s. The energy per metre that the robot learned from its legs on this day was only about 0.005 Wh per metre, because most of the energy is used while the robot stands or waits rather than while it drives. In this 70-minute session the charge shown fell from 40 % to 9 %, about 8 Wh of the 26 Wh usable, so a full pack lasts roughly 3.5 to 4 hours of operation.

F. Network Effects and User Interface

The robot also recorded the Wi-Fi delay between itself and the laptop. With the phone hotspot in the crowded expo hall, the average round-trip time was 533 ms and 27.9 % of the probes were lost, whereas with a router in the afternoon it was 22 ms and 70 ms on the two maps used, with at most 1.5 % of the probes lost. In the two houses it was 108 ms and 182 ms. Every wheel command goes from the laptop to the robot over Wi-Fi. The poor network in the morning was probably one reason for the lost positions and the failed motor restarts. The battery, the map and the crowd also changed between morning and afternoon, so the causes cannot be separated. Fig. 10 shows how a delivery is explained to the sender on the website: each wait, narrow passage, detour and recovery appears as a short sentence with its time.

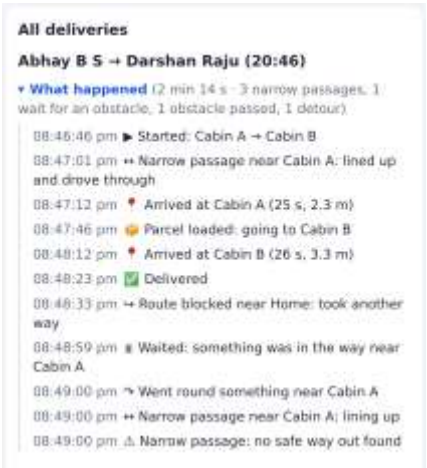

Fig. 10. "What happened" timeline of a delivery on the web app

V. DISCUSSION

The robot was built from parts costing about Rs. 28,200, and it completed most of its deliveries in two homes and in a crowded hall. The recovery layer was not an extra for rare cases: in 44 % of the completed legs the robot had to wait, manoeuvre through a gap or find its position again, and it still reached the goal. We did not repeat the routes with the layer switched off, so we cannot say how many of these legs would have failed without it; we can only say that before the layer existed, each of these situations needed a person to walk over.

The 78 % success rate is higher than the 60 % reported for a simple college document robot [12], but the two numbers come from different buildings, routes and failure rules, so they show the general level and not an exact difference. Our confidence interval (61–89 %) is also wide, because the number of jobs is small.

Most failures came from the surroundings and the power supply rather than from path planning: a tight corner with no room to turn, a crowd with no free route, a weak battery and a congested Wi-Fi network. The robot handled most of these safely by bringing the parcel back, going home or asking for help. Both motor faults in November came late in the session, at about 28 % and 20 % battery, which suggests that low voltage plays a part, but two faults are too few to be sure.

The work has clear limits. The laptop is in the control loop, so a poor network directly affects driving; running the navigation on the robot would remove this. The lid is closed by the servo but has no latch. The ultrasonic sensor misses round or angled objects, and the 2D LiDAR does not see objects below its scan plane. The battery percentage comes from the voltage, corrected for the load current, and not from counting the charge, so the prediction is approximate, and only its decisions were tested, not how close the predicted energy was to the energy used. The energy model counts driving distance and stops but not the time the robot is switched on, which uses most of the energy; a model with an energy-per-minute term would predict better. The recovery thresholds were tuned by hand, and the software changed between test days. Finally, the evaluation covers one robot over five days without ground-truth positions, so "recovered" means that the robot's own scan-to-map check passed.

VI. CONCLUSION

This paper presented a low-cost, map-aware indoor delivery robot on ROS 2 and Nav2, with a web app for users, a servo-closed parcel lid, autonomous mapping, and a self-recovery layer that handles lost position, blocked routes, narrow gaps and motor faults inside the delivery job. In 32 evaluable real deliveries the robot completed 78 %, recovered its position without help in 41 cases with a median of 4 s, and ended 70 of 80 gap manoeuvres without help, while needing a person on 4 of 112 legs. In a later session it delivered all 16 jobs that were not cancelled or stopped on purpose, recovered from all 7 deliberate kidnaps, 5 of them without help, and handled all 3 low-battery trials safely. The main limitations are the small number of jobs, a single robot without ground-truth positions, the laptop and Wi-Fi in the control loop, and a battery estimate that is only approximate. Future work will move the navigation onto the robot, add a laser time-of-flight sensor and a camera for low and moving obstacles, relocate the drive wheels to the middle of the body, add a latch to the lid, add an energy-per-minute term to the battery prediction, and run a longer trial in the college library and office.

ACKNOWLEDGEMENT

We thank our guide, Dr. Darshan B D, and the Department of Electronics and Communication Engineering, SJB Institute of Technology, Bengaluru, for their support, and our families and friends who allowed us to test the robot in their homes.

REFERENCES

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

[2] S. Macenski, F. Martín, R. White, and J. Ginés Clavero, "The Marathon 2: A navigation system," in Proc. 2020 IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS), 2020, pp. 2718–2725, doi: 10.1109/IROS45743.2020.9341207.

[3] S. Macenski and I. Jambrecic, "SLAM Toolbox: SLAM for the dynamic world," Journal of Open Source Software, vol. 6, no. 61, Art. no. 2783, 2021, doi: 10.21105/joss.02783.

[4] B. Yamauchi, "A frontier-based approach for autonomous exploration," in Proc. 1997 IEEE Int. Symp. Computational Intelligence in Robotics and Automation (CIRA'97), 1997, pp. 146–151, doi: 10.1109/CIRA.1997.613851.

[5] Y. Liu, Y. Lu, C. Peng, Q. Bu, Y. Liang, and J. Sun, "Autonomous vehicle based on ROS2 for indoor package delivery," in Proc. 2023 28th Int. Conf. Automation and Computing (ICAC), 2023, pp. 1–7, doi: 10.1109/ICAC57885.2023.10275227.

[6] T. Kim, G. Kang, D. Lee, and D. Shim, "Development of an indoor delivery mobile robot for a multi-floor environment," IEEE Access, vol. 12, pp. 45202–45215, 2024, doi: 10.1109/ACCESS.2024.3381489.

[7] M. Ibrahim, M. Salem, I. Hossain, A. Sarbabidya, and S. Saha, "Autonomous robot navigation with SLAM, AMCL, and real-time user interaction for task management and identity verification," in Proc. 2024 IEEE 3rd Int. Conf. Robotics, Automation, Artificial-Intelligence and Internet-of-Things (RAAICON), 2024, pp. 252–257, doi: 10.1109/RAAICON64172.2024.10928347.

[8] S. Cui, M. Chen, L. He, Y. Zhang, M. Sun, and X. Duan, "Design of mobile robot interaction system based on SLAM particle filter," in Proc. 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conf. (IAEAC), 2021, pp. 93–97, doi: 10.1109/IAEAC50856.2021.9390996.

[9] V. Mhadeshwar et al., "QR-code guided autonomous medical assistant robot for indoor hospital environments," in Proc. 2025 3rd World Conf. Communication & Computing (WCONF), 2025, pp. 1–5, doi: 10.1109/WCONF64849.2025.11233784.

[10] S. Shrestha, S. Parajuli, P. Shah, B. Shrestha, and M. Guragai, "Navigation of mobile robot with Nav2 and SLAM using LiDAR," in Proc. 2025 4th Int. Conf. Sentiment Analysis and Deep Learning (ICSADL), 2025, pp. 579–585, doi: 10.1109/ICSADL65848.2025.10933029.

[11] K. Patil, M. Patil, G. Kawade, K. Bait, and A. Mane, "Lidar-guided autonomous campus delivery robot using Raspberry Pi and ROS 2," International Journal for Multidisciplinary Research, vol. 8, no. 2, Art. no. 73356, 2026, doi: 10.36948/ijfmr.2026.v08i02.73356.

[12] A. Akinwumi, R. Elewa, S. Monye, I. Okokpujie, and A. Banjoko, "Development of a door-to-door document delivery robot for a college in an academic environment," WSEAS Transactions on Design, Construction, Maintenance, vol. 5, pp. 64–75, 2025, doi: 10.37394/232022.2025.5.7.

[13] N. Hawes et al., "The STRANDS project: Long-term autonomy in everyday environments," IEEE Robotics & Automation Magazine, vol. 24, no. 3, pp. 146–156, 2017, doi: 10.1109/MRA.2016.2636359.

[14] R. Valner et al., "Scalable and heterogenous mobile robot fleet-based task automation in crowded hospital environments—a field test," Frontiers in Robotics and AI, vol. 9, Art. no. 922835, 2022, doi: 10.3389/frobt.2022.922835.

[15] Y. Lee et al., "Feasibility of autonomous medication delivery robots considering elevator utilization in high-traffic hospital environments," Digital Health, vol. 12, Art. no. 20552076261437181, 2026, doi: 10.1177/20552076261437181.

[16] Z. Ismail and I. Bukhori, "Efficient detection of robot kidnapping in range finder-based indoor localization using quasi-standardized 2D dynamic time warping," Applied Sciences, vol. 11, no. 4, Art. no. 1580, 2021, doi: 10.3390/app11041580.

[17] N. Akai, "Reliable Monte Carlo localization for mobile robots," Journal of Field Robotics, vol. 40, no. 3, pp. 595–613, 2023, doi: 10.1002/rob.22149.

[18] M. Zhang, L. Ma, Y. Wu, K. Shen, D. Huang, and H. Leung, "Tackling the kidnapped robot problem via sparse feasible hypothesis sampling and reliable batched multistage inference," IEEE Transactions on Instrumentation and Measurement, vol. 75, Art. no. 7504614, 2026, doi: 10.1109/TIM.2026.3694741.

[19] S. Du, T. Chen, Z. Lou, and Y. Wu, "A 2D-LiDAR-based localization method for indoor mobile robots using correlative scan matching," Robotica, vol. 43, no. 2, pp. 514–541, 2025, doi: 10.1017/S026357472400198X.

[20] A. De Luca, L. Muratore, and N. G. Tsagarakis, "Autonomous navigation with online replanning and recovery behaviors for wheeled-legged robots using behavior trees," IEEE Robotics and Automation Letters, vol. 8, no. 10, pp. 6803–6810, 2023, doi: 10.1109/LRA.2023.3313052.

[21] D. Bozhinoski et al., "MROS: Runtime adaptation for robot control architectures," Advanced Robotics, vol. 36, no. 11, pp. 502–518, 2022, doi: 10.1080/01691864.2022.2039761.