UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Tecnura

https://revistas.udistrital.edu.co/index.php/Tecnura/issue/view/1329

DOI: https://doi.org/10.14483/22487638.23211

INVESTIGACIÓN

Picking up Recycled Paper and Package Delivery System Indoors using a Mobile Robot

Sistema de Recogida de Papel de Reciclaje y Despacho de Paquetes en Interiores usando un Robot Móvil

Jorge Enrique Caicedo Martínez[1], Bladimir Bacca Cortes[2]

Fecha de Recepción: 6 de febrero de 2025 **Fecha de Aceptación**: 8 de septiembre de 2026

Cómo citar: Caicedo Martínez, J.E. y Bacca Cortés, E.B., «Picking up Recycled Paper and Package Delivery System Indoors using a Mobile Robot», *Tecnura*, vol. 30, n.° 90, oct. 2026. 4–28. https://doi.org/10.14483/22487638.23211

Abstract

Context: To produce 1-kg paper, 324 L of water is needed [1], so recycling paper is a better option.
Objective: This work describes the development of a system for picking up recycled paper and a package delivery system indoors.
Methodology: The system includes a web page to request recycled paper pick-up and package delivery, a SQL database to track these requests, a mobile robot with ROS to execute the requests, and a multi-goal path planning system that heuristically solves the traveling salesman problem (TSP).
Results: To validate this work, integration and field tests were performed. Integration tests confirmed that the functional requirements were satisfied, and field tests observed that the system responds linearly with respect to time to user requests with an average correlation coefficient of 0.9591.
Conclusions: The REM software tool presented in this work is a valid option for implementing a recycled paper pickup or package delivery service system using mobile robots that consider the Lin-Kernighan heuristic to solve the multi-goal path planning problem.
Keywords: Pickup and delivery, mobile robot, travel salesman problem.

1 He received his bachelor's degree in Electronics Engineering from the Universidad del Valle, Cali, Colombia in 2019. He also received his master's degree at the same institution in 2023. He is currently working at Abka Colombia S.A.S., Cali, Colombia. Email: jorge.caicedo.martinez@correounivalle.edu.co

2 He received his bachelor's degree in Electronics Engineering from the Universidad del Valle, Cali, Colombia in 1999. Subsequently, he pursued his master's degree at the same institution in 2004. Then he received his Ph.D. degree on Technology from the Universitat de Girona, Spain in 2012. Now, he is a Full Professor at Universidad del Valle, Cali, Colombia. His main research interests are mobile robotics, simultaneous localization and mapping systems, and computer vision. Email: bladimir.bacca@correounivalle.edu.co

Resumen

Contexto: Para producir 1kg de papel se necesitan 324 L de agua [1], por lo tanto reciclarlo es una mejor opción.
Objetivo: Este trabajo describe el desarrollo de un sistema para la recogida de papel y el despacho de paquetes en interiores.
Metodología: El sistema incluye una página web para la solicitud de recogida de papel de reciclaje y despacho de paquetes, una base de datos SQL para su seguimiento, un robot móvil con ROS para la ejecución de las peticiones y un sistema de planificación de trayectorias multiobjetivo que resuelve heurísticamente el problema del agente viajero (TSP).
Resultados: La validación de este trabajo se realizó con pruebas de integración y de campo. Las pruebas de integración validan el cumplimiento de los requerimientos funcionales; y en las pruebas de campo se observó que el sistema responde linealmente con respecto al tiempo ante la demanda de peticiones con un coeficiente de correlación promedio de 0.9591.
Conclusiones: La herramienta de software REM presentada en este trabajo es una opción válida para implementar un sistema de servicio de recogida de papel reciclado/entrega de paquetes mediante robots móviles que considera la heurística de Lin-Kernighan para resolver el problema de planificación de rutas multiobjetivo.
Palabras clave: Recogida y entrega, robot móvil, problema del agente viajero.

Introduction

Actually 70 % of the current negative environmental impact on Latin America is due to logging for products such as timber, paper, and pulp [1]. An interesting tool to compute the environmental impact of paper in different presentations is The Paper Calculator [2], which is a peer-reviewed work. It addresses the environmental impacts of the paper life cycle; for instance, 1 ton of paper produced has the following impacts on the environment: 4 tons of wood used, 33 million BTUs, 2.5 tons of greenhouse gases, and 22.8 gallons of water, among others.

Universidad del Valle, like many other institutions, is at the end of the paper production chain, but it is concerned about paper waste. In [4], Universidad del Valle consolidates the institutional policy of solid waste management. This document shows the following data: the total paper and cardboard generated in the Melendez campus is 8,63% (7,555.9 kg / month), and in the San Fernando campus is 12,73% (968.5 kg / month). This means that our institution generates 102 metric tons of paper and cardboard per year. It is worth noting that regarding recycling, or at least facilitating paper collection, 1 ton of newsprint saves 1 ton of wood, while recycling 1 ton of office paper saves approximately 2 tons of wood [3].

Then, the pick-up of recycled paper tasks can be implemented in indoor office environments, avoiding people from traveling to a collection point, since most of the time people are working and focused on their business. In addition, it is interesting to use the same platform for the delivery of goods. To do so, this work proposes a system composed of two main software modules as depicted in Figure 1: first, a frontend web software which receives the user requests and handles the deliveries; and, a backend software executed by the mobile robot using ROS [5] which performs the pick-up and the delivery tasks. The frontend WEB software differentiates between pickup and delivery requests; pickup requests are

performed by users from their workplaces, and delivery requests are performed by one administrative user at the collection center. The frontend then sends these requests to the backend using JSON data objects through ROS bridge. At the backend, these requests are treated as multiple goal positions to be visited by the mobile robot. To implement this multi-goal path planning the Lin-Kernighan heuristic [6] was integrated to the ROS navigation stack. The solution described in this work also considers events such as battery level of the mobile robot, the number of items stored in the container placed on the mobile robot and the requests schedule.

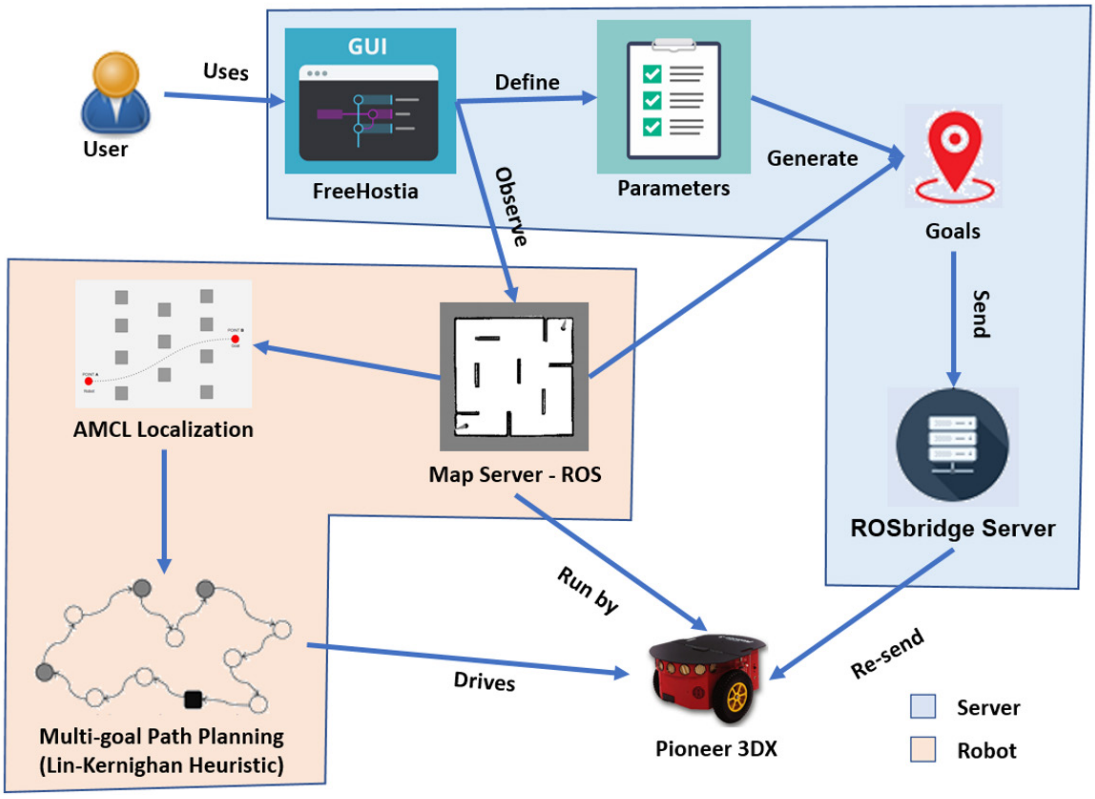


Figure 1. *Concept diagram of REM.*

Source: Authors.

This document is structured as follows: related works are described in the next section; the design and implementation of REM are described in the methodology section; then this document finishes with the tests and results section, and the conclusions and future work section.

Related Works

Table 1 shows the most relevant works to this paper. These relevant works are compared considering their application context (App.), sensors used, environment where the robot is deployed (Env.),

navigation or planning method used (Nav. and Planning), type of map used, and what type of features the robots use to navigate.

Table 1. *Related works comparison*

Ref.	App.	Sensors	Env.	Nav.	Planning	Map	Nav. Feat.
[7]	Aut. mail delivery	2D Lidar RGBD	Indoors and outdoor	VLNS-T	VLNS-T	Real-world	Vision and range.
[8]	Pickup / delivery	RGB	Indoors aerial	Visual servoing	Visual servoing	N/A	Object segment.
[9]	Pickup / delivery	RGB, GPS	Outdoor, aerial	Visual servoing	CPP	N/A	Visual odometry
[10]	Pickup / delivery	RGB	Outdoor	SIPP	ILP	3D graph	Visual descriptors
[11]	Aerial inspection	RGB, weather sensors	Outdoor	A*	Decision tree, 2-step heuristic	Weather level map	Weather data, geo-position.
[12]	Pickup / delivery	2D lidar	Indoors, outdoors	ARNL	Lin-Kernighan	Occupancy grid	Range, heading
[13]	Pickup / delivery	2D lidar	Indoors	Hybrid TA, A*	Central, LKH-3	Occupancy grid	Range, heading
[14]	Multiple Pickup / delivery	2D lidar	Indoors	A*	Genetic algorithm	Topologic	Range, heading
[15]	Autonomous part Inspection	2D lidar	Indoors	Roadmap / A*	Cristofides heuristic	Occupancy grid	Range, heading
[16]	Multiple Pickup / delivery	2D lidar	Indoors	A*	HDSF, prima dual	Topologic	Range, heading
[17]	Hospital logistics	2D lidar	Indoors	A*	Nav. Time / Task heuristic	Occupancy grid	Range, heading
[18]	Warehouse	N/A	Indoors	N/A	Genetic algorithm	N/A	Travel / task time
[19]	Automatic painting	GPS	Outdoor aerial	Dubins path	Lin-Kernighan	N/A	Waypoints
[20]	N/A	2D lidar	Indoor	DWA, A*	Ant Colony	Occupancy grid	Range, heading
[21]	Multiple Pickup / delivery	2D lidar	Indoor	N/A	LNS-PBS	Occupancy grid	Range, heading
[22]	Retail vehicle	2D lidar	Indoor	N/A	Ant colony opt.	Occupancy grid	Range, heading
[23]	NA	NA	Indoors and outdoor	GA-VNS	GA-VNS	Topologic	NA
[24]	NA	2D Lidar	Indoors	DWA	ACO	Occupancy grid	Range, heading

Table 1 shows that the most commonly used sensors to assist mobile robots in local navigation tasks are cameras [11], [10], [9], [8], [7] and LIDAR [7], [12], [13], [14], [15], [16], [17], [20], [21], [22]. This is due to the large amount of information that images can provide to extract many features using depth images and image textures. Range information is used for its relative ease of use and is employed for multiple tasks (obstacle detection, positioning correction on maps). In [19], GPS is used to estimate the state of the robot, but due to the way this type of sensor operates, it is only useful outdoors. In [23] and [18], no sensors are mentioned because those works focused on the efficient handling of pickup/delivery goals using custom navigation and planning strategies.

Another aspect to consider is the working environment. Most of the works found deal with outdoor applications, and works such as [11], [10], [13], [19] deal with applications using drones. Although using drones in indoor applications is not impossible (as seen in [10]), these works take advantage of the freedom of movement provided by the ability to move freely in space, which is easier in outdoor environments. In addition, most indoor applications [9], [8], [7], [14], [15], [16], [17], [18], [24] use ground robots.

As for navigation strategies between pick-up/drop-off points, these are closely related to the type of map to be used and the available sensors; for example, in [9], [8], [14], [15], [16], [24], they use occupancy grids and navigation algorithms that leverage the information provided by the current map and the range information to trace trajectories (Dijkstra, ARNL or A*). However, in [9] a couple of particular algorithms are used that use the execution time of the journey as a basis, forming a complete route plan based on the available goals. In the case of [11], [10], which do not use maps, the presence of cameras is used to plan routes using visual servoing, which uses the local relative position of a visible target as a basis to plan and control the movements necessary to reach the target. And there is also the case of [17], which uses topological maps even though the TSP implementation was developed in a manipulator, which also shows that this problem is not limited to autonomous moving vehicles. It is observed that all the works that deal with the traveling salesman problem are works that must plan routes that consider multiple points.

The works presented in [7], [11], [10], and [16] are focused on the fundamental problem of pickup and delivery, concentrating on giving the robot the ability to deliver objects to a goal in the first place [11], [10]. Or focused on complementary functions, such as load transfer [7] or on providing a service with customizable requests that are served in an order defined mainly by priority or delivery time, so that route planning is not performed immediately for all tasks [16].

Since, in this project, navigation is only part of the problem, it should also look at path planning strategies that consider multiple pick-up/drop-off points at the same time. In most cases, heuristics are used for planning trajectories to drop-off points, among which are the Lin-Kernighan heuristic [8], [14], [19]. It consists of the temporary substitution of pairs of sub-routes to create a new shorter route, or the

Christofides heuristic [15], which ensures solutions within a factor of 1.5 of the optimal solution. Although, there are works that use more than one heuristic, such as [7] that uses multi-vehicle heuristics, 2- or 4-approximation heuristics, and meta-heuristics for multiple-goal planning. There are algorithms that take advantage of the field of view of cameras for path planning, such as servo positioning [10] and CPP (Coverage Path Planning), a path planning algorithm that divides the work area into several convex sections, each explored by a robot in a zig-zag pattern [11]. These tasks are more critical in works that focus on the implementation of TSP because they must consider routes to multiple points at the same time, plan routes with different starting points, and select the routes that together produce the most efficient total trajectory [8], [14], [15]. Another different approach is proposed by [24], where authors involve ant colony optimization algorithms to solve multi-target path planning; here, the DWA algorithm is also used for local planning. However, in [24] authors show results of limited indoors environments without clear application.

As for the features used for robot navigation, the use of range information predominates, [9], [8], [7], [14], [15], followed by visual and range odometry [11], [10], and points of interest (in multiple forms) [13], [7]. Other useful features can be found in works that use specialized cameras, such as depth cameras [7], or cameras for the detection of the relative position of objects in the scene [10] and texture [11].

Methodology

This section shows the development of REM, which was conceived considering the RUP software engineering methodology [25]. This methodology includes the following deliverables: functional and non-functional requirements, conceptual diagrams, use cases, sequence diagrams, class diagrams, source code, and test reports. However, due to space constraints, this section describes functional and non-functional requirements, the ROS node diagram, and the GUI description.

Design of the Robotic Enterprise Messenger (REM)

Considering all the properties described in the related works, the functional and non-functional requirements of REM are described as follows:

Functional Requirements: a friendly Web GUI where users can perform their requests, define the type of request (pickup of recycled paper or package delivery), and visualize the environment map, the robot path and the progress of the task. The mobile robot must localize itself and plan its multi-goal target points autonomously. The system must receive different types of requests dynamically. The mobile robot builds a map based on a LiDAR sensor. The mobile robot will attract the user's attention the moment it arrives at a pickup/delivery point. Upon accomplishing all requests, the mobile robot must return to the home location autonomously.

Non-Functional Requirements: the development environment is ROS Kinetic; the mobile robot platform is a Pioneer3DX; the minimum Android OS version for the portable device is 8.0; the environment map is built previously using GMapping; the database server is MySQL 10.2; the frontend was developed using NodeJS 20.2.

The development of REM uses ROS of the backend application in the mobile robot and communicates between the frontend and the backend through a ROS bridge. ROS applications can be modeled using graph diagrams; however, the backend of REM was developed using object-oriented programming, and the class diagram is shown in Figure 2. The entry point to the backend is the *goal_mid* class, where requests for recycled paper pickup or package delivery tasks are received from the frontend. It then translates the message information to the *maploc* class which handles publishing all required information to other ROS nodes in the mobile robot. Most of the work in the backend is done in the *goal_proc2* class, which organizes all the requests received by *goal_mid* to perform the multi-goal path calculations using the Lin-Kernighan heuristics and send the goals progressively to the ROS navigation stack. The *robot_dismiss* class deals with the reception of recycled paper or picking up the packages at user point. Finally, the *robot_position* and *robot_map* classes deal with holding the position and map of the environment and sending these data to the frontend for visualization.

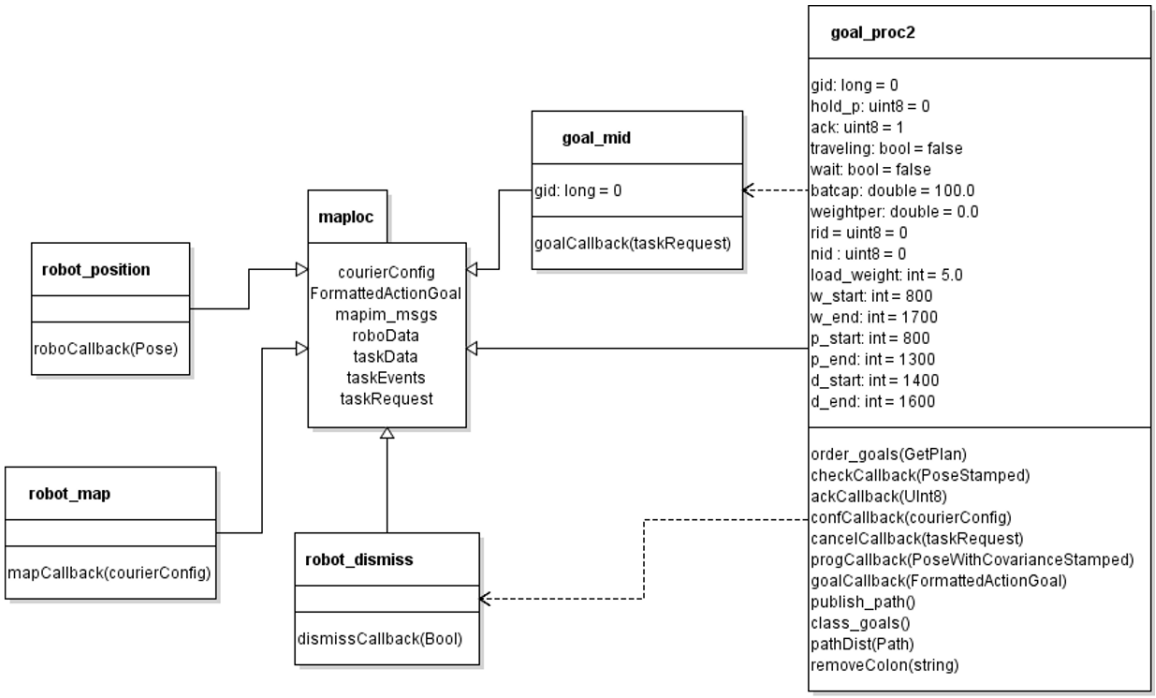


Figure 2. *Class diagram of the backend application in the mobile robot.*

Source: Authors.

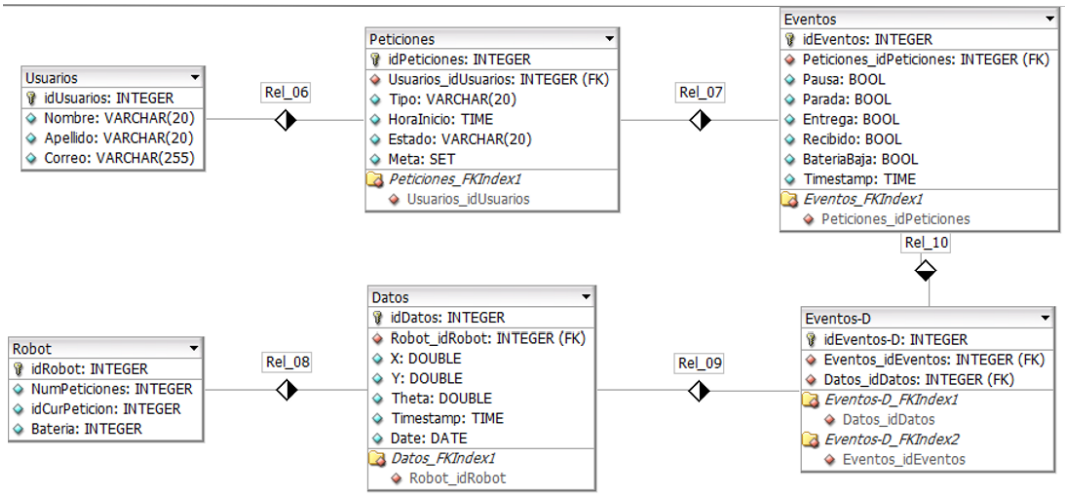


Figure 3. *Relational model of the database.*

Source: Authors.

The REM software tool uses a MySQL relational database, the model of which is shown in Figure 3. The logic behind this model is as follows: users perform one or more requests to the system; at each request many events can happen; also, robots in charge of performing the requests generate positional data, and this data is related to events in a particular request.

Implementation of the Robotic Enterprise Messenger (REM)

To better comprehend how REM works, a top-bottom approach is described below. The REM frontend available for users who perform pickup of recycled paper requests and administrators who perform package delivery requests is shown in Figure 4. The REM frontend starts by asking users for their access credentials, that is: username (email) and password as shown in Figure 4a. In case the users have forgotten their password, a recovery page is shown to users as depicted in Figure 4b. This page sends a new password to the registered email account. In addition, new users can start a registration procedure shown in Figure 4c. Here, the users must enter their first name, last name, email and password. Then, registered users can start to perform requests as shown in Figure 4d. In this Figure, users who want to perform a pickup of recycled paper request can click on the red marks which correspond to where the user is placed; or, if the user is an administrator, this type of user can perform a package delivery request and the destination of this delivery is the red marks in Figure 4d. In addition, in Figure 4d users can also log-out, or if it is an administrator user, this user can configure the daily schedules for pickup of recycled paper and package delivery as shown in Figure 4e.

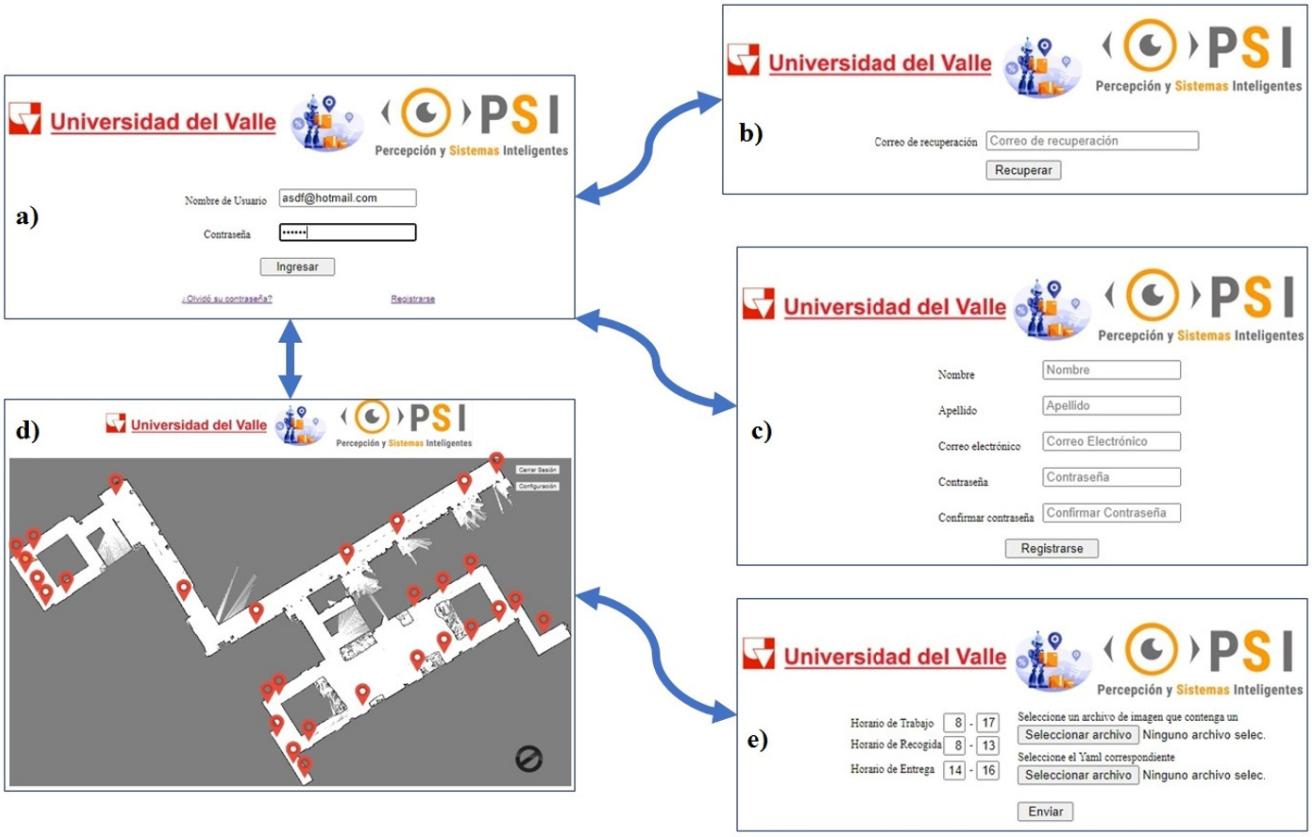


Figure 4. *Navigation diagram of the user frontend.*

Source: Authors.

The pickup of recycled paper and package delivery requests performed by users in the frontend are saved in the database and sent to the mobile robot through ROS bridge [26]. Also, data are sent by the mobile robot to the frontend and the database using the ROS bridge. Figure 5 shows the interaction between the frontend and the websocket server, and the mobile robot. The following type of information is sent from the frontend to the mobile robot: pickup of recycled paper / package delivery requests, and configuration parameters (i.e. schedules). Then, ROS bridge converts these data to valid ROS topics in the mobile robot (*/new_pickup_goal* and */courir_config*). In turn, the mobile robot sends pose data (X, Y and orientation), and status information (battery level, number of pending requests, current request in execution, the map of the environment, and the current path to execute) to the frontend. All this information is packed in JSON objects by ROS bridge and exchanged with the frontend. In the frontend, all data are stored in the database for tracking purposes.

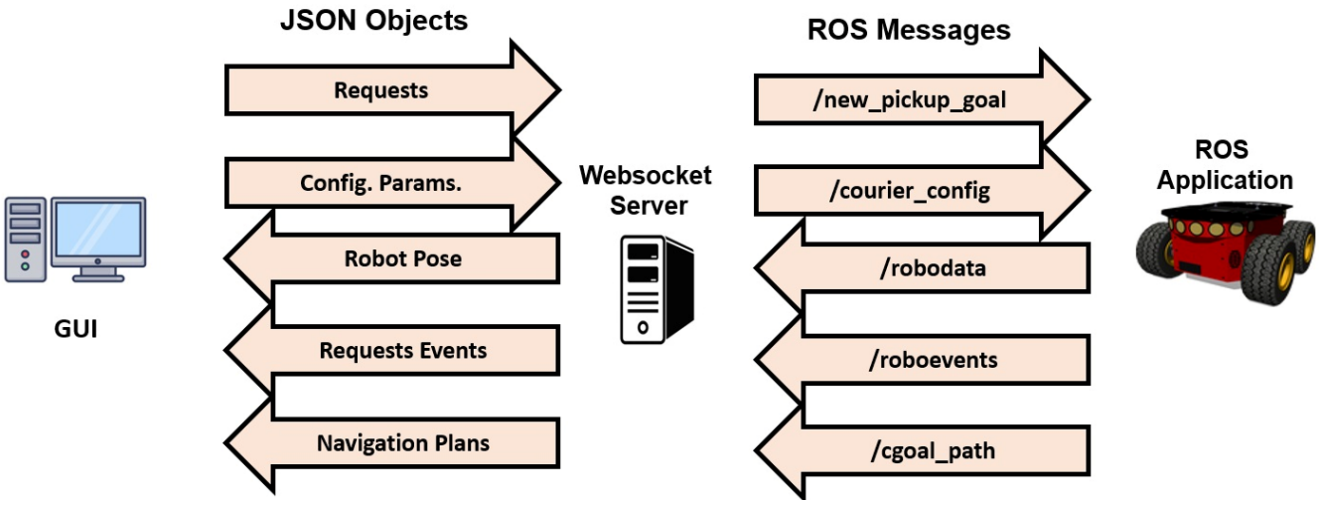


Figure 5. *Frontend and backend communication.*

Source: Authors.

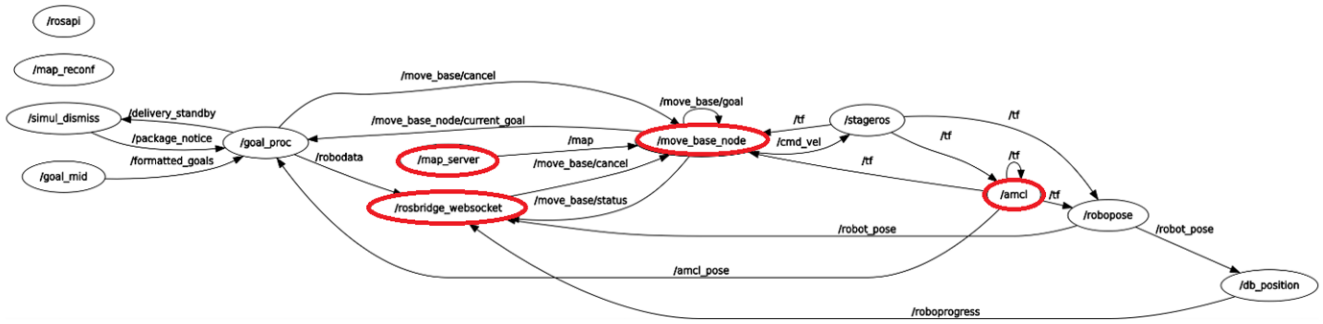


Figure 7. *ROS nodes of REM in the mobile robot.*

Source: Authors.

In the backend, ROS messages which contain the new goal requested and the configuration parameters are processed by the */proc_goal* node as depicted in Figure 7. The */proc_goal* node has the implementation of the Lin-Kernighan heuristic to deal with multi-goal navigation. To do that it supports its functionality with the */move_base_node* and */amcl* nodes. These nodes belong to the ROS navigation stack to guarantee the mobile robot's localization and the point-to-point path planning using a previously computed map. */robotpose* and */db_position* are nodes that format all the information needed to be sent to the frontend through the */rosbridge_websocket* node. This enables the frontend to display the current position of the robot in the map and the progress of the current goal.

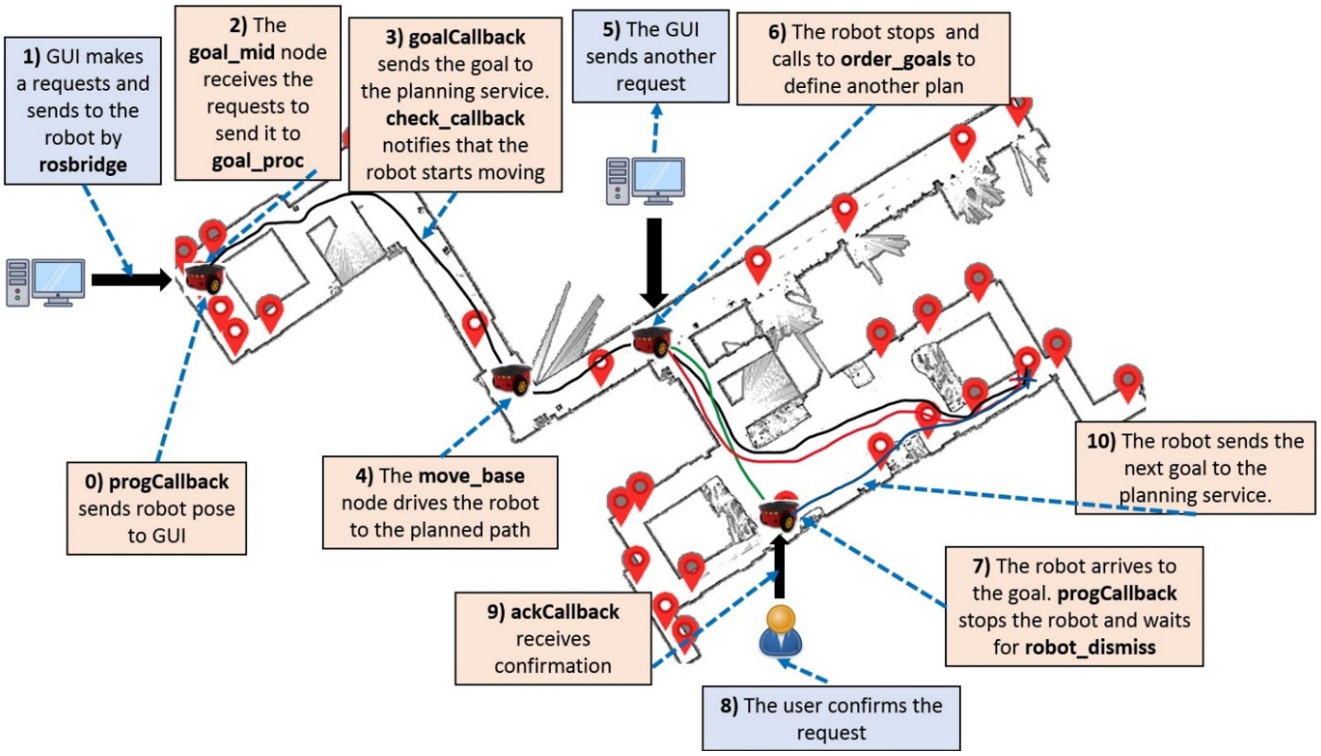


Figure 8. *Process diagram example.*

Source: Authors.

To understand the pipeline process in a recycled paper pickup / package delivery application, Figure 8 shows how the REM software tool deals with these requests. Blue boxes in Figure 8 are related to the frontend, and the others to the backend in the mobile robot. The pipeline process is described as follows (in this case to process two user requests):

1. At the very beginning, the mobile robot and backend software send the current pose to the GUI and database system.
2. Users access the Web page described in Figure 4 to make a pickup of recycled paper / package delivery request. This request is sent to the backend in the mobile robot using ROS bridge.
3. The user request is received in the backend software based on ROS by the */goal_mid* node as depicted in Figure 7. Which, in turn, preprocesses the information sent by the GUI and resends the request to the */goal_proc* node.
4. Next, in the */goal_proc* node this new request is added to the current list of goals to visit in order to give this data to the Lin-Kernighan heuristic algorithm, which returns the planned path.
5. Then, the */goal_proc* node manages the different goals to visit returned by the Lin-Kernighan heuristic algorithm, and each goal is sent in turn to the ROS navigation stack using the */move_base_node*. At

this point the mobile robot starts moving, the position of the robot is computed by the */amcl* node and sent to the GUI through the ROS bridge connection.

6. The REM software tool was developed to capture any other requests at any time. These requests are asynchronous and can happen at any time. For instance, at this moment, other users can perform other pickup/delivery requests, which follow the same steps as described in 1.
7. The new request is received again by */goal_mid* node. Then, the mobile robot stops computing the new plan using the Lin-Kernighan heuristic algorithm implemented in the */goal_proc* node.
8. When the new plan is computed, the mobile robot can arrive at the first goal of the new plan. The robot arrives at the goal location, then it notifies the user of this event through the GUI. And the robot waits for the robot to dismiss events through the */robot_dismiss* topic.
9. The user is aware of the arrival event and activates the robot dismissal event using an industrial push button installed on the mobile robot.
10. The robot dismiss event is sent through the ROS bridge and received by the REM frontend software.
11. Then, the mobile robot using the */goal_proc* node sends the next goal in the path to the ROS navigation stack until there are no more goals to visit.
12. Finally, the mobile robot travels to the home location whenever there are no requests to process.

The process described above shows an ideal execution of the events. However, REM software tool also considers the following alternative events and how to deal with them:

Low battery level: In this case the */goal_proc* node stops the robot, suspends the current path planning, notifies the REM frontend of this event (which saves pending goals as incomplete), and sends the *home* location as next goal to reach to the ROS navigation stack.

Mobile robot waiting for long time: In case the robot arrives at the goal location, but the user does not respond to the request, the */goal_proc* node starts a timeout of 2 minutes. At the end of this time the system automatically generates the robot dismiss event, but this request is stored in the database as not attended.

End of the recycled paper pickup schedule: In case of reaching the end of the recycled paper pickup schedule, the mobile robot does not receive more recycled paper pickup requests, but finishes the current planned goals.

No route to gal: In case the mobile robot faces dynamic obstacles not previously mapped that block the path of the mobile robot, and it is not possible to reach the goal, this event is stored in the database, and the mobile robot continues with the next goal planned.

Tests and Results

To validate the functionality of the REM software tool, two types of tests were performed: first, integration tests to confirm that the functional requirements were satisfied; and second, a field test to evaluate how the REM software tool behaves considering the number of requests sent versus the time to accomplish as documented in [27].

Hardware Description

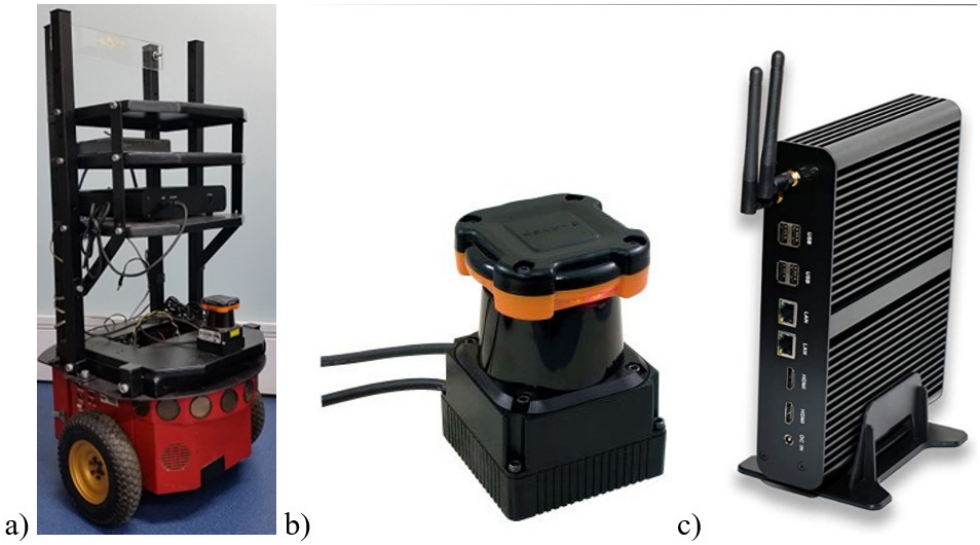


Figure 9. *a) Pioneer 3DX mobile robot. b) Hokuyo UTM-30LX laser sensor. c) Mini-PC Kingdel.*

Figure 9 shows the hardware used in this work. In Figure 9a the Pioneer3DX mobile robot can be observed, which has a structure where the Mini-PC Kingdel is placed, the communication router and the industrial push button for user feedback are installed. This mobile robot has a total payload of 17Kg, a maximum speed of 1.2m/s, maximum angular speed of 300°/s, and an autonomy of 8h. The main sensor of the REM software tool is a Hokuyo UTM-30LX laser shown in Figure 9b. Which can work indoors and outdoors, it has a maximum range of 30m and a FOV (Field of View) of 270°. The Kingdel Mini-PC has an Intel Core i7-5500U processor at 2.4Ghz, 16GB of RAM, and a 1TB hard disk. It has Ubuntu Linux 16.04, ROS Kinetic, OpenCV 2.9.17 and Aria 2.8.

Map Creation

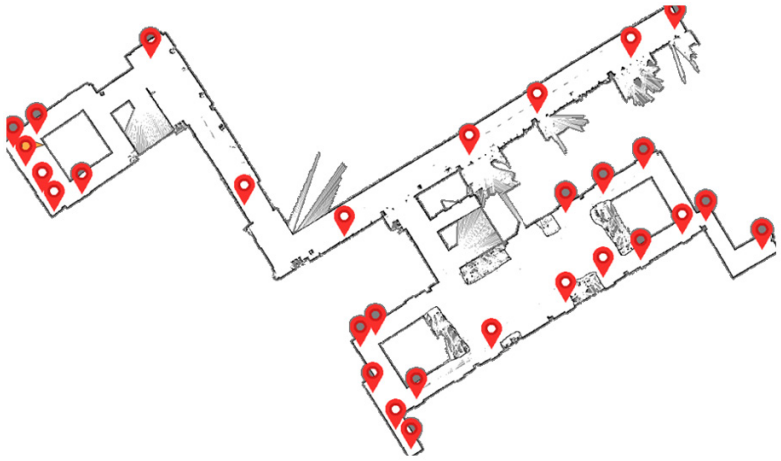


Figure 10. *Map of the environment.*

Before performing the integration and field tests, it is mandatory to compute the map of the environment. To do so, the following steps are followed:

1. A ROS package was created to include data acquisition from the Hokuyo laser sensor, the motion interface to the robot using the *rosaria_client* node, and the *gmaping* node to build the map.
2. Then, using the teleoperation node for the Pioneer 3DX, the mobile robot was driven across the second floor of E52 and E53 buildings of the School of Electrical and Electronic Engineering of the Universidad del Valle.
3. At the moment of returning to the starting position, the computed map was saved (See Figure 10).
4. Using an image processor such as GIMP, or similar applications, many interesting places were tagged, for instance: teacher offices, electronic laboratories, different research laboratories and computer rooms.
5. Finally, using the tags' image coordinates, each place was introduced into the database, in this way the REM frontend can show the pickup of recycled paper / package delivery places to users.

Integration Tests

To validate if the REM software tool satisfies the functional requirements, three integration tests were considered: first, sending requests between the frontend and the backend in the mobile robot; second, the path planning process; and third, the robot's dismissal when the mobile robot arrives to the goal. These tests were documented using the RUP engineering software methodology. Then, Tables 2, 3 and 4 show the information considered for each test. All tests were performed considering the Pioneer 3DX mobile robot, internet access, and a PC; and the REM frontend and backend software, and ROS Kinetic.

Table 2. *Integration test No. 1: sending requests between the frontend and the backend in the mobile robot.*

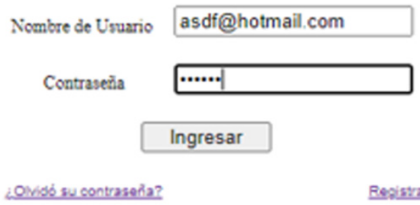
Functional requirements involved: • The GUI should allow access to the GUI's main page (where requests are made). • The GUI should be able to save the information defined in JSON objects. • The GUI should store the request information in a database. • ROS should be able to receive and extract the information from JSON objects. • ROS should be able to publish the information extracted from the JSON objects in the appropriate ROS topics.	
Test Data	User email, user password, JSON object
Procedure	1. Start the REM backend in ROS Kinetic and in the home position. 2. Using a PC accesses the main page of REM frontend. 3. Using the user 's email and password to access the pickup of recycled paper / package delivery request page. 4. Send a pickup of recycled paper / package delivery request.

Expected Result

1. The user can access the pickup of recycled paper / package delivery request page using the user email and password.
2. When the user requests a pickup of recycled paper / package delivery task a confirmation pop-up.
3. When the request is sent, the ROS topic /formatted_goals show the goal information packed in JSON objects.

Obtained Result

1. User access:



2. Request page and confirmation pop-up.



3. Formatted goal after receiving the JSON object in the backend, and ready to use in ROS.

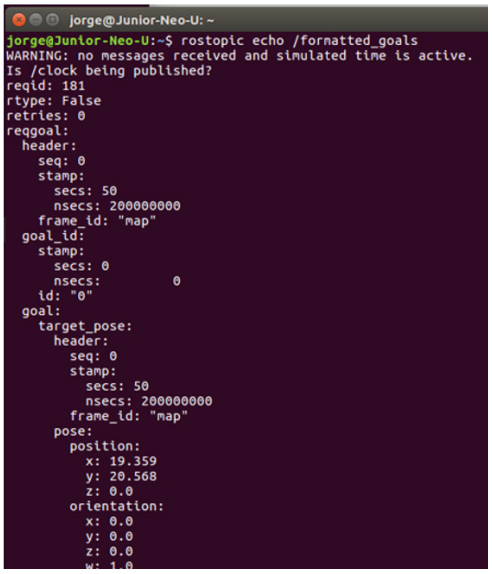


Table 3. *Integration test No. 2: path planning process.*

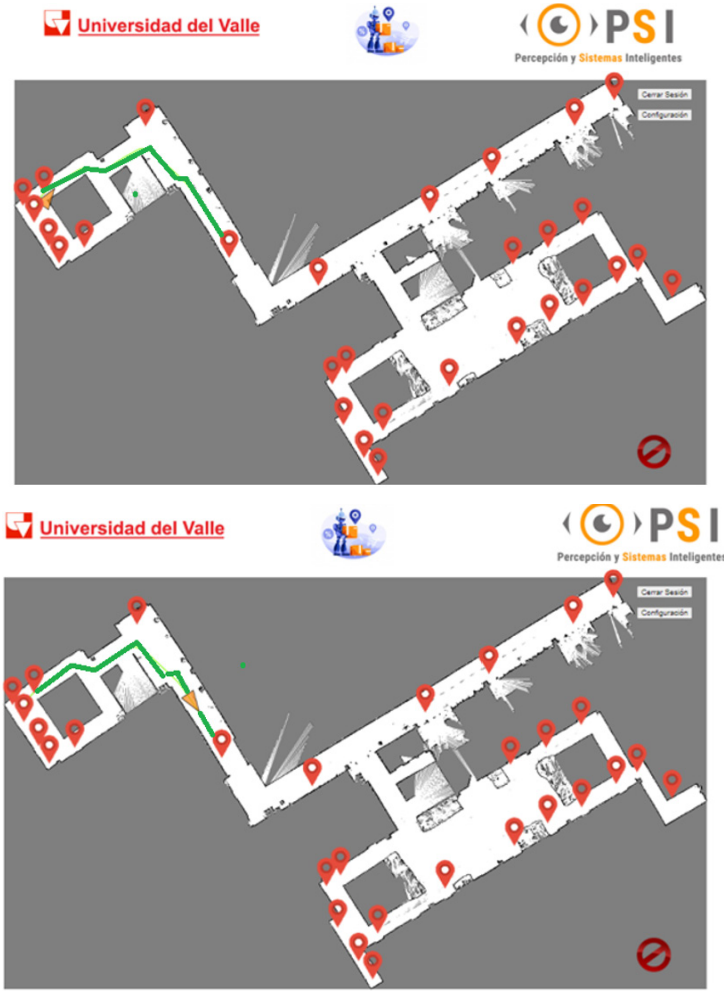
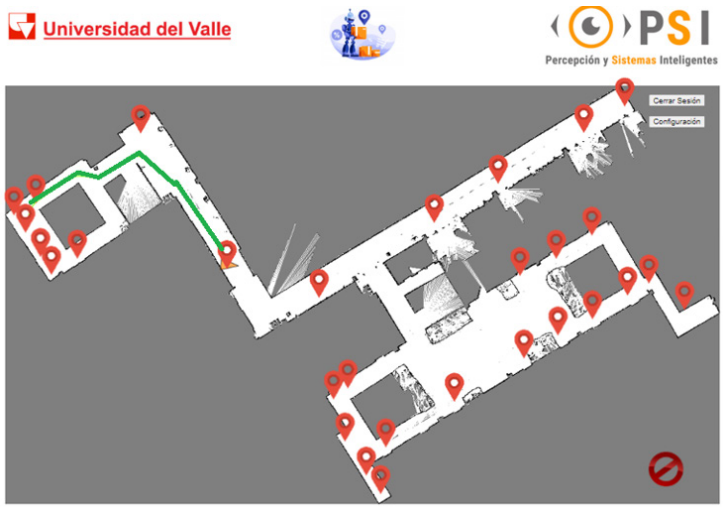
Functional requirements involved:	The mobile robot must localize itself and plan its multi-goal target points autonomously. The system must receive different types of requests dynamically. To visualize the environment map, the robot path and the progress of the task.
Test Data	**The map of the environment, the formatted goals, and the mobile robot position.**
Procedure	1. Waiting for a user request in the /formatted_goal ROS topic. 2. Include the received goal in the list of goals to visit and compute the robot path using the Lin-Kernighan heuristic. 3. Send the goals in the path plan to /move_base node. 4. Moving the robot and waiting for its arrival to the goal.
Expected Result	1. The JSON object arrives with the mobile robot. 2. A robot plan is computed using the Lin-Kernighan heuristic. 3. The mobile robot sends the path and location to the frontend for tracking. After receiving the JSON object, the /goal_proc ROS node plans the robot path (green line in Figures) and starts moving the robot (orange triangle in Figures) using the /move_base node.
Obtained Result	

Table 4. *Integration test No. 3. The robot dismisses when the mobile robot arrives at the goal.*

Functional requirements involved:	
The mobile robot will attract the user's attention the moment it arrives at a pickup point for recycled paper / package delivery point. The mobile robot will dismiss and continue its path.	
Test Data	The map of the environment, the mobile robot's position, and the robot's request to the user.
Procedure	1. The mobile robot arrives at the goal. 2. The mobile robot attracts the user's attention and waits. 3. The user activates the industrial push-button and dismisses the mobile robot.
Expected Result	1. The mobile robot arrives at the goal, the REM backend communicates this event, and the REM frontend shows this event to the user. 2. The robot is dismissed; this event is registered in the database.
Obtained Result	1. The mobile robot arrives at the goal and alerts the user.  2. The user activates the industrial push button and dismisses the robot. This can be observed in the database. 

Field Test

The field tests performed in this work were conducted considering the localization and path planning parameters described in Table 5. This table shows the parameters that differ from their default values.

Table 5. *Localization and path planning parameters.*

Localization		
Param. Name	**Value**	**Description**
laser_max_beams	30	Laser beams considered for AMCL.
min_particles	500	Minimum number of particles for AMCL.
max_particles	5000	Maximum number of particles for AMCL.
initial_pose_x	2.0	Initial X pose of the robot in the map.
initial_pose_y	25.0	Initial Y pose of the robot in the map.
initial_pose_a	0.0	Initial heading of the robot in the map.
DWA Local Planner		
Param. Name	**Value**	**Description**
max_vel_x / min_vel_x	0.65 / 0.1	Maximum and minimum linear speed.
max_rotational_vel	1.0	Maximum rotational speed.
xy_goal_tolerance	0.1	XY tolerance on achieving goal.
yaw_goal_tolerance	0.05	Yaw tolerance on achieving goal.
sim_time	2.5	Forward time to simulate trajectories.
sim_granularity	0.05	Granularity for the trajectory simulation
Local Cost Planner		
Param. Name	**Value**	**Description**
rolling_window	True	New obstacles are added around the robot pose.
footprint_padding	0.1	Safety margin in meters around the robot
width / height	3.0 / 3.0	Size of the local cost map
resolution	0.08	Resolution of the local map.
Global Cost Planner		
Param. Name	**Value**	**Description**
footprint_padding	0.1	Safety margin in meters around the robot
Costmap Common Params.		
Param. Name	**Value**	**Description**
obstacle_range	3.0	Max. distance to be used to add obstacles
footprint	[[0.35, -0.30], [0.35, 0.30], [-0.40, 0.30], [-0.40, -0.30]]	Robot shape to be used for the local and global planners.
inflation_radius	0.4	Distance from obstacles

The quantitative field test includes seven individual tests, ranging from 5 to 20 pickups of recycled paper / package delivery requests. In all, these tests the following conditions were satisfied:

- The time of goal completion and the total time spent across all tests was measured.
- The goal requests were added dynamically after the mobile robot started moving.
- The selection of the goal to request was random, but only considering those goals within 10m of the current position.
- To simulate the mobile robot, dismiss the event: when the mobile robot arrives at the goal, it stops, waits for 10s, and automatically proceeds to the next goal.

This type of test is common in multi-goal path planning systems, as reported in [26], [27], and [28]. Since each test requires high energy consumption, they were performed using full battery levels.

Figure 11 shows the results of the field tests performed. Over these tests, the wider time variations were observed when the random goal requested was far from the current mobile robot position. Which implies travelling on a bigger non-Euclidean path with respect to other closer goals. Of course, it is an expected result. Despite this fact, all test curves have similar linear behavior.

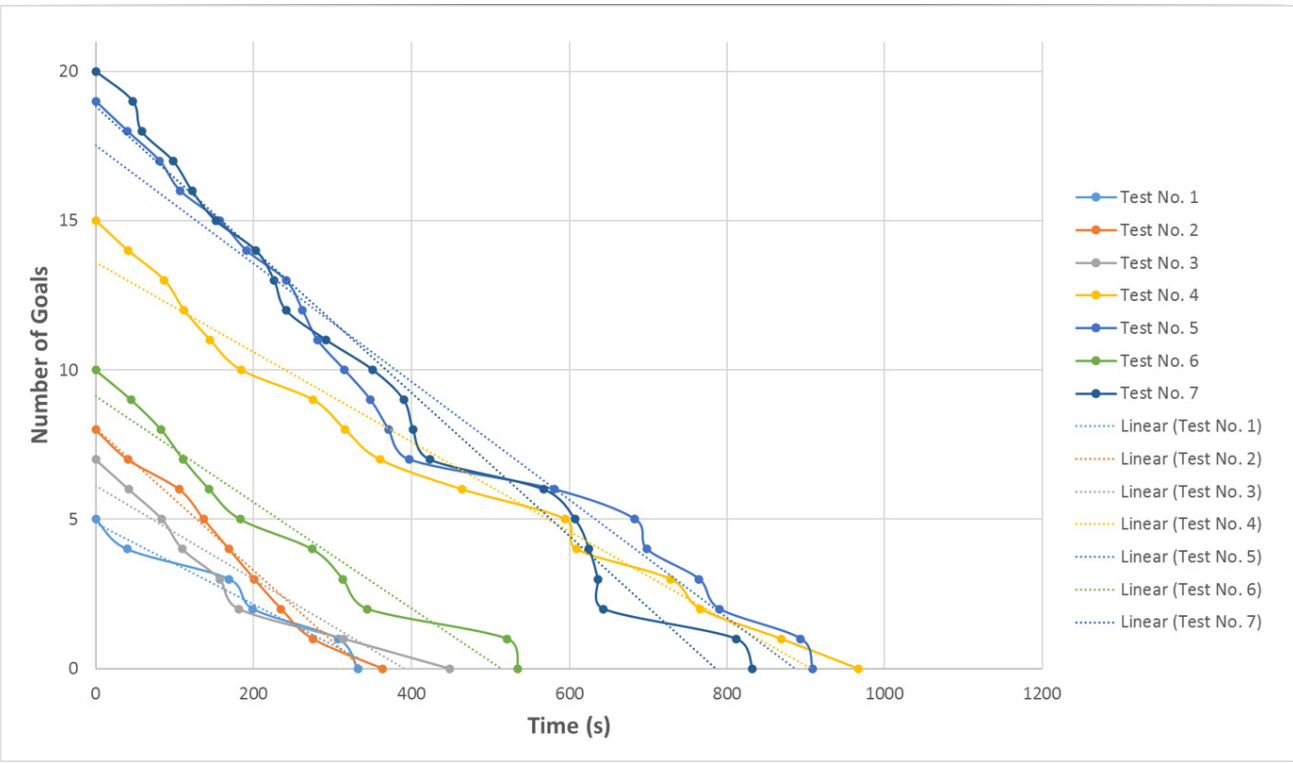


Figure 11. *Plot of the number of goals reached vs Time.*

Source: Authors.

To check this linear behavior, the correlation coefficient of each dataset was computed. Table 5 shows the correlation coefficients of all tests performed. Using these coefficients, a linear plot was generated, which can be observed in Figure 11.

Table 6 shows that the system proposed in this work has a predictable trend in relation to the final fulfillment time of the recycled paper pickup / package delivery service. This is a desirable feature in a service robot for recycled paper pickup / package delivery tasks.

Table 6. *Test correlation coefficients.*

Test No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Corr. Coef.	0,968432	0,900171	0,98071	0,951945	0,974711	0,960929	0,977096

Additional field tests were performed to verify the REM behavior in the presence of blocked road and low battery alerts. In normal operation, REM uses static and dynamic maps to perform multi-goal path planning as described in previous sections; however, roads can be temporarily blocked due to maintenance tasks, people, or the moving of furniture. In this case, it is expected that REM re-plan the robot trajectory if possible.

Figure 12 shows the behavior of REM under the presence of a blocked road. Figure 12a shows the path planning result in normal conditions; here, the green square shows the initial position of the mobile robot, the purple square shows the goal target, and the planned trajectory is drawn in green. This experiment was repeated by physically blocking the upper road (blue square); as a result, REM re-plans the trajectory to the goal using the lower part of the map.

Figure 13 shows the result of having a low battery in the middle of the multi-goal path planning task. Figures 13a to 13c show the response of REM when there are three goals to visit; the right part of each figure shows the battery level. In Figure 13d, a fourth goal is added; however, a low battery alert is issued (observed in the blue square); as a result, the fourth goal is rejected and REM automatically guides the mobile robot to the home location for recharging.

Considering the related works presented above, the proposed REM system, and the tests performed, the main contributions of this work are as follows: first, a pickup and delivery system capable of operating in a multi-goal scenario and responding linearly with respect to the number of requests; second, a REM software tool which integrates a multi-goal handler based on [6], a backend developed with ROS, and a frontend development for user management; and finally, REM was tested in a real indoor environment in the presence of people and considering a physical space of 100m x 150m.

As shown in Table 1, few works were tested in real environments or in simulated ones with spacious areas [7], [17], [15], [11], [14], [13], [24]. For instance, comparing the results achieved in [13] and [24], these the authors explore solutions to deal with multi-agent pickup and delivery in simulation environments [13], or using ant colony optimization combined with dynamic window approach algorithms to solve the multi-target problem [24]. However, while the environment covers a wide area of work, it is simulated, there is no evidence of using a standard framework such as ROS in there is no integration with a front end for user interaction. In [14], authors propose a patrolling solution with actually uses ROS on Matlab, and simulated environments, not real ones; however, there is no front-end GUI for user interaction.

In contrast, in [7], the authors propose a Very Large Neighborhood Search with Transfers algorithm tested in spacious indoor areas; however, that development does not use the ROS framework, which hinders the replicability of results, and there is no front-end GUI system for user interaction. In [17], the authors propose a robot base for delivery tasks in hospital environments; however, again, the authors do not use a development framework, nor do they present a front-end GUI to interact with users. Finally, in [11], the authors propose an interesting multi-goal UAV for large-scale inspection system, and they include a GUI for user interaction; however, they do not use ROS as a framework.

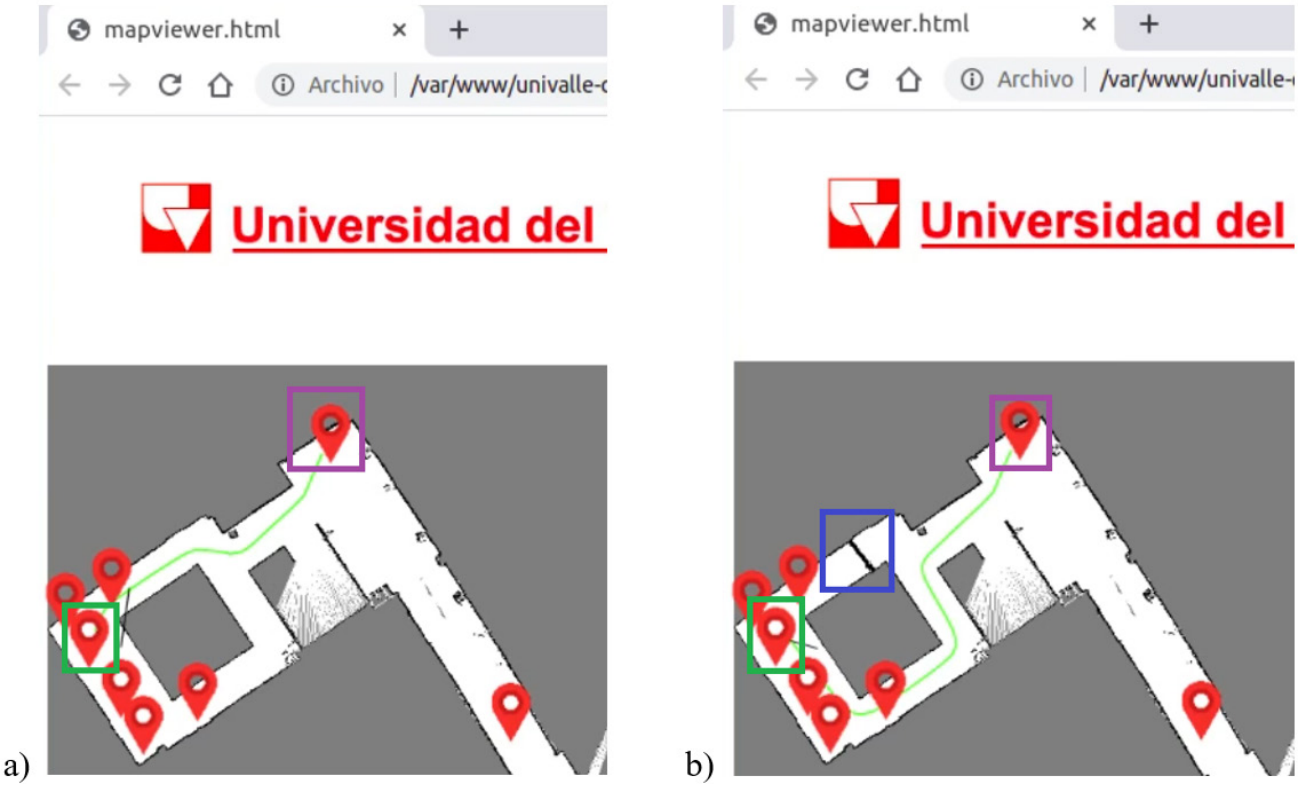


Figure 12. *a) Path planning with an unblocked road. b) Path planning with a blocked road.*

Source: Authors.



Figure 13. *a) Going to the first goal. b) Going to the second goal. c) Going to the third goal. d) Adding a fourth goal, which is rejected due to a low battery level.*

Source: Authors.

Conclusions

This work describes the REM software tool, which is proposed to implement a recycled paper pickup and package delivery service using indoor mobile robots. The REM software tool is composed of two parts: first, a frontend where users can submit recycled paper / package delivery requests; these requests are saved in a SQL database, sent to the mobile robot using a ROS bridge, allowing users to track the progress of their requests via the interface by observing the map and the robot's current position; and second, a backend that runs on the mobile robot. The backend is able to process multiple-goal requests dynamically, compute the robot's path plan using the Lin-Kernighan heuristic integrated into ROS, drive the robot to each individual goal using the ROS navigation stack, attract the user's attention when the robot arrives at a goal, and perform a dismissal process to continue to the next goal.

To validate the functionality of REM software tool, two different tests were performed: first, integration tests to prove that REM software tool satisfies the functional requirements; and second, a field test to show how the REM software tool behaves when users perform different numbers of requests.

The integration tests show evidence that REM software tool satisfies the basic functional requirements proposed in this work. However, this type of service mobile robot task must deal with multiple goal requests, which are performed dynamically while the mobile robot is in motion. To observe how the REM software tool responds to this scenario, the field test included seven different situations where the mobile robot attended to 5 to 20 recycled paper pickup or package delivery requests, which were presented to REM software tool dynamically. As a result, the time vs. number of goals reached behaved linearly, which is desirable in this kind of system. In addition, low battery and blocked road tests were performed, as a consequence, REM responded as expected. In case of low battery alerts, REM guides the mobile robot to the home position, rejecting new goal requests; in case of blocked road alerts, REM replans the mobile robot's trajectory. Thus, the REM software tool presented in this work is a valid option for implementing a recycled paper pickup or package delivery service system using mobile robots that consider the Lin-Kernighan heuristic to solve the multi-goal path planning problem.

References

[1] Cartly, "The Impacts of Paper on The Environment." [Online]. Available: https://cartlyapp.com/en/the-impacts-of-paper-on-the-environment/. Accessed: Feb. 27, 2024.

[2] Environmental-Paper-Network, "The Paper Calculator." [Online]. Available: https://c.environmentalpaper.org/. Accessed: Mar. 04, 2024.

[3] A. Pérez, *Sistema de gestión integral de residuos sólidos ordinarios (SGIRS)*. Cali, Colombia, 2020

[4] Open-Robotics, "ROS: Home." [Online]. Available: https://www.ros.org/. Accessed: Sep. 06, 2022.

[5] K. Helsgaun, "An effective implementation of the Lin-Kernighan traveling salesman heuristic," *Eur. J. Oper. Res.*, vol. 126, no. 1, pp. 106–130, 2000, doi: https://doi.org/10.1016/S0377-2217(99)00284-2

[6] B. Coltin, *Multi-Agent Pickup And Delivery Planning With Transfers*. Dissertations, no. May, 2014, doi: https://doi.org/10.1109/ICRA.2014.6907709

[7] A. Gawel *et al.*, "Aerial picking and delivery of magnetic objects with MAVs," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 5746–5752, 2017, doi: https://doi.org/10.1109/ICRA.2017.7989675

[8] R. Hahnemann, D. Schindler, M. Kamel, R. Siegwart, and J. Nieto, "A decentralized multi-agent unmanned aerial system to search, pick up, and relocate objects," in *SSRR 2017 - 15th IEEE Int. Symp. Saf. Secur. Rescue Robot. Conf.*, 2017, pp. 123–128, doi: https://doi.org/10.1109/SSRR.2017.8088150

[9] B. Araki, J. Strang, S. Pohorecky, C. Qiu, T. Naegeli, and D. Rus, "Multi-robot path planning for a swarm of robots that can both fly and drive," *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 5575–5582, 2017, doi: https://doi.org/10.1109/ICRA.2017.7989657.

[10] P. Oettershagen, J. Förster, L. Wirth, J. Ambühl, and R. Siegwart, "Meteorology-Aware Multi-Goal Path Planning for Large-Scale Inspection Missions with Long-Endurance Solar-Powered Aircraft," Nov. 2017

[11] K. Hernandez, B. Bacca, and B. Posso, "Multi-goal Path Planning Autonomous System for Picking up and Delivery Tasks in Mobile Robotics," *IEEE Lat. Am. Trans.*, vol. 15, no. 2, pp. 232–238, Feb. 2017, doi: https://doi.org/10.1109/TLA.2017.7854617.

[12] M. Liu, H. Ma, J. Li, and S. Koenig, "Task and Path Planning for Multi-Agent Pickup and Delivery," in *Proc. 18th Int. Conf. Auton. Agents MultiAgent Syst. AAMAS*, 2019, doi: https://doi.org/10.65109/QTKR1570

[13] A. Koubâa, O. Cheikhrouhou, H. Bennaceur, M. F. Sriti, Y. Javed, and A. Ammar, "Move and Improve: a Market-Based Mechanism for the Multiple Depot Multiple Travelling Salesmen Problem," *J. Intell. Robot. Syst. Theory Appl.*, vol. 85, no. 2, pp. 307–330, 2017, doi: https://doi.org/10.1007/s10846-016-0400-x

[14] M. Ulrich, G. Lux, L. Jürgensen, and G. Reinhart, "Automated and Cycle Time Optimized Path Planning for Robot-Based Inspection Systems," *Procedia CIRP*, vol. 44, pp. 377–382, 2016, doi: https://doi.org/10.1016/j.procir.2016.02.021

[15] J. Bae and W. Chung, "Efficient path planning for multiple transportation robots under various loading conditions," *Int. J. Adv. Robot. Syst.*, vol. 16, no. 2, p. 172988141983511, Mar. 2019, doi: https://doi.org/10.1177/1729881419835110

[16] C. A. Acosta Calderon, E. R. Mohan, and B. S. Ng, "Development of a hospital mobile platform for logistics tasks," *Digit. Commun. Netw.*, vol. 1, no. 2, pp. 102–111, Apr. 2015, doi: https://doi.org/10.1016/j.dcan.2015.03.001

[17] A. Oliinyk, I. Fedorchenko, A. Stepanenko, M. Rud, and D. Goncharenko, "Implementation of Evolutionary Methods of Solving the Travelling Salesman Problem in a Robotic Warehouse," in *Springer*, Cham, 2021, pp. 263–292, doi: https://doi.org/10.1007/978-3-030-43070-2_13

[18] A. Tkachev, *Multi-Goal Path Planning for Spray Writing with Unmanned Aerial Vehicle*. Czech Technical University in Prague, 2020

[19] B. Zhang, W. Jin, X. Gao, and W. Chen, "A Multi-Goal Global Dynamic Path Planning Method for Indoor Mobile Robot," in *2021 3rd International Symposium on Robotics & Intelligent Manufacturing Technology (ISRIMT)*, IEEE, Sep. 2021, pp. 97–103, doi: https://doi.org/10.1109/ISRIMT53730.2021.9597044

[20] Q. Xu, J. Li, S. Koenig, and H. Ma, "Multi-Goal Multi-Agent Pickup and Delivery," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Oct. 2022, pp. 9964–9971, doi: https://doi.org/10.1109/IROS47612.2022.9981785

[21] A. A. H. Qizilbash, A. K. Yadav, K. Bregler, and W. Kraus, "Ant Colony Optimization for Retail Based Capacitated Vehicle Routing Problem with Pickup and Delivery for Mobile Robots," in *2023 European Conference on Mobile Robots (ECMR)*, IEEE, Sep. 2023, pp. 1–6, doi: https://doi.org/10.1109/ECMR59166.2023.10256346

[22] F. Gao, X. Guo, X. Zheng, and S. Ning, "Research on AGV Path Planning of Simultaneous Pickup and Delivery with Time Window," in *2023 26th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, IEEE, May 2023, pp. 1778–1782, doi: https://doi.org/10.1109/CSCWD57460.2023.10152796

[23] L. Huang *et al.*, "Path Planning for Delivery Robots Based on an Improved Ant Colony Optimization Algorithm Combined with Dynamic Window Approach," *Sensors*, vol. 26, no. 72, pp. 1–30, 2026, doi: https://doi.org/10.3390/s26010072

[24] Taryana Suryana, "Rational Unified Process (RUP)," *Ration. Unified Process RUP*, vol. 3, no. September, pp. 1–6, 2007

[25] J. Mace, J. Lee, R. Toris, B. Alexander, D. Bertram, and M. Gruhler, "RosBridge Protocol Specification," GitHub. [Online]. Available: https://github.com/RobotWebTools/rosbridge_suite/blob/groovy-devel/ROSBRIDGE_PROTOCOL.md. Accessed: Sep. 01, 2018

[26] C. H. Wurll, D. O. Henrich, and H. E. Wörn, "MULTI-GOAL PATH PLANNING FOR INDUSTRIAL ROBOTS," in *International Conference on Robotics and Application (RA'99)*, Santa Barbara, USA, 1999, pp. 1–7

[27] J. Janos, V. Vonasek, and R. Penicka, "Multi-Goal Path Planning Using Multiple Random Trees," *IEEE Robot. Autom. Lett.*, vol. 6, no. 2, pp. 4201–4208, Apr. 2021, doi: https://doi.org/10.1109/LRA.2021.3068679

[28] L. Hu and Z. Kong, "Multi-target point path planning algorithm for mobile robot based on probabilistic roadmap," *Intell. Serv. Robot.*, vol. 19, no. 1, p. 9, Jan. 2026, doi: https://doi.org/10.1007/s11370-025-00670-6