การประชุมวิชาการทางวิศวกรรมไฟฟ้า ครั้งที่ 48
The 48th Electrical Engineering Conference (EECON-48)
วันที่ 19-21 พฤศจิกายน 2568 ณ โรงแรมฟูราม่า จังหวัดเชียงใหม่

Energy-Aware Navigation for Unmanned Ground Vehicles: A Dijkstra-Based Approach on Raspberry Pi 5

Rahut Puttharaksa[1], Nattavit Piamvilai[1*],Suriyotai Supanyapong[1], Prem Tooboonlong[1], and Polatip Thongpet[1]

[1*]Department of Electrical Engineering Technology, College of Industrial Technology,
King Mongkut's University of Technology North Bangkok,
E-mail: nattavit.p@cit.kmutnb.ac.th*

Abstract

This paper presents an Energy-aware Navigation system designed for embedded platforms using Dijkstra's algorithm implemented on a Raspberry Pi 5. The system combines real-time GPS input, OpenStreetMap-based mapping, and battery State of Charge (SoC) monitoring to compute energy-efficient paths while minimising total Energy Consumption. When energy is insufficient to reach the destination, the system reroutes through the nearest charging station. All Route Planning computations are performed locally on the Raspberry Pi 5, with lightweight processing requirements suitable for real-time use on low-cost hardware. The system enables intelligent navigation decisions based on power availability by embedding energy constraints directly into the routing process. This approach highlights the potential of compact, flexible embedded systems to support adaptive path planning in energy-limited environments. Furthermore, the system provides a modular foundation for future expansion, including integrating advanced sensors, energy modeling enhancements, and context-aware routing strategies for mobile platforms.

Keywords: Dijkstra's algorithm, Energy Consumption, Energy-aware Navigation, GPS, Raspberry Pi 5, Route Planning

1. Introduction

With the rise of vehicle technologies, modern navigation systems increasingly rely on GPS, digital maps, and shortest-path algorithms to improve transportation efficiency. These systems aim to minimise travel distance, time, and energy usage. Integrating route planning with real-time energy data is essential for energy-aware decision-making as energy becomes a critical concern, especially for electric vehicles.

A simulation system was developed using a custom vehicle model with Raspberry Pi 5 as the core processor to address this. The system begins with a web interface where users select a destination, while the starting point is obtained via GPS. Battery SoC is detected through onboard sensors. This data is sent to the Pi, which uses the OSMnx library to convert OSM data into a weighted graph locally [1]. Dijkstra's algorithm is then applied to compute the shortest, most energy-efficient path.

If the estimated energy is insufficient, the system reroutes to the nearest charging station. The route is displayed via an interactive map, allowing users to view energy metrics and path details through a browser or onboard screen.

Simulations confirm that the system computes energy-aware routes and adapts to battery status, two test cases were examined one with sufficient energy, and one without. In both, predicted energy use matched actual values. The system also achieved consistently low computation times, confirming its capability for real-time operation on embedded hardware.

2. Methodology

2.1 Overview of the Energy-Aware Navigation System Using Dijkstra's Algorithm

This navigation system is built upon Dijkstra's algorithm, which computes the shortest and most energy-efficient path within a graph-based road network. The system comprises five core components: a web-based user interface, Raspberry Pi 5 as the onboard processor, a vehicle equipped with a GPS module, a central server for data management, and OSM data. as shown in Fig. 1. illustrates the overall architecture and the interaction between these components, showing how each part collaborates to enable real-time, energy-aware navigation with minimal hardware resources.

Firstly, the user selects a destination through the interface, while the Raspberry Pi 5 retrieves the vehicle's current location in real time using GPS data. The OSM data is pre-processed via the OSMnx library and stored locally on the Raspberry Pi as a weighted graph, where nodes and edges represent intersections and road segments with associated distance and energy cost attributes.

When a route request is triggered, the Raspberry Pi loads the graph and executes Dijkstra's algorithm to compute the optimal path by minimising cumulative costs, such as travel distance or estimated energy usage. The system then calculates the total trip energy consumption and compares it to the remaining battery capacity. If the available energy is insufficient, it dynamically reroutes through the nearest charging station to ensure the trip's completion.

Real-time GPS updates and SoC monitoring enable the system to adapt to changing travel and battery conditions. This ensures continuous, efficient, and energy-aware pathfinding throughout the journey, even under constrained power availability.

*Corresponding Author

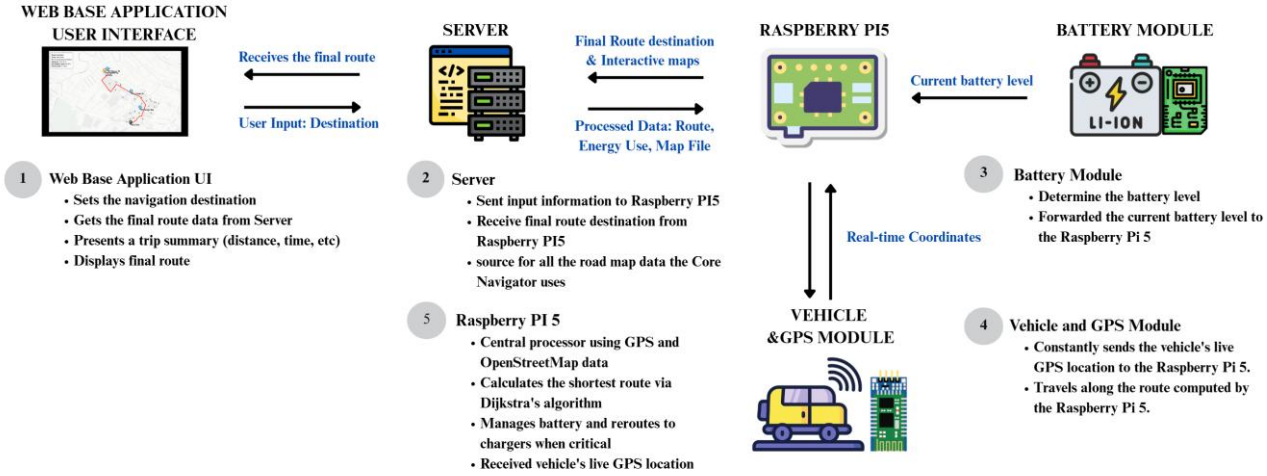

Fig. 1. Overview of the proposed energy-aware navigation system architecture based on real-time GPS and Dijkstra's algorithm

2.2 Pathfinding Process Using Dijkstra's Algorithm

The core of the navigation system's pathfinding logic is Dijkstra's Algorithm, a foundational and highly efficient method for determining the shortest path between two nodes in a weighted graph. The algorithm systematically explores the graph, guaranteeing the discovery of the most optimal route based on cumulative distance.

The operational flow of The algorithm is demonstrated in the visualisation provided as shown in Fig. 2. In this specific example, the process begins at the designated start node (A) and is set to find the shortest path to the destination node (F).

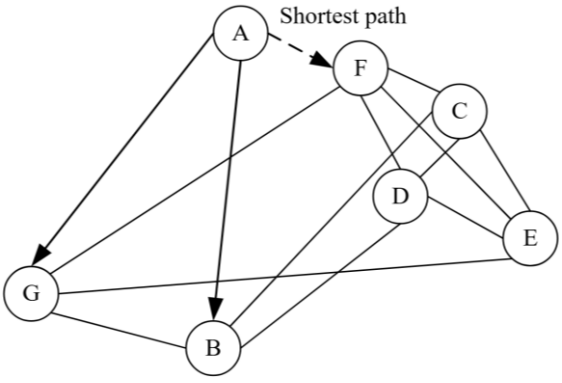

Fig. 2. Visualisation of the Dijkstra's algorithm in progress, showing visited nodes and the current path exploration from Start Node A to Destination Node F

Dijkstra's Algorithm is a well-established graph traversal method used to determine the shortest path between two nodes in a weighted graph. The process begins with the initialisation step, where the source node is assigned a distance of zero. In contrast, all other nodes are given an initial value of infinity to signify that they are unreachable at the start. These nodes are then placed into a priority queue that helps maintain the order based on the smallest known distance.

The Algorithm then proceeds into its iterative exploration phase. In each iteration, it extracts the unvisited node with the minimum known distance, marking it as the currently processing node. Once a node is processed—meaning all of its neighbors have been examined-it is marked as visited, signifying that the shortest path to that node has been finalised.

The core mechanism of the Algorithm lies in the edge relaxation step. For each neighbor of the current node, the Algorithm calculates the potential new distance by summing the current node's distance with the weight of the connecting edge. If this new value is smaller than the previously recorded distance for that neighbor, it is updated accordingly as shown in Eq. (1)

$$\text{If } d(u) + w(u,v) < d(v) \\ \Rightarrow d(v) := d(u) + w(u,v) \tag{1}$$

where

$d(u)$ = Current known shortest distance from the source to node u

$w(u,v)$ = Weight between node u and node v

$d(v)$ = Current known distance to node v

This process continues until the destination node is reached. At that point, the algorithm terminates and reconstructs the path by backtracking from the destination to the source, identifying the optimal route. This behavior is illustrated as shown in Fig. 2, where node A is the starting point and the shortest path is visually traced to the destination. The dashed line represents the finalized route.

Driven by a greedy strategy, Dijkstra's algorithm consistently prioritizes the most efficient paths, offering a reliable and computationally effective solution for route planning in vehicular navigation system.

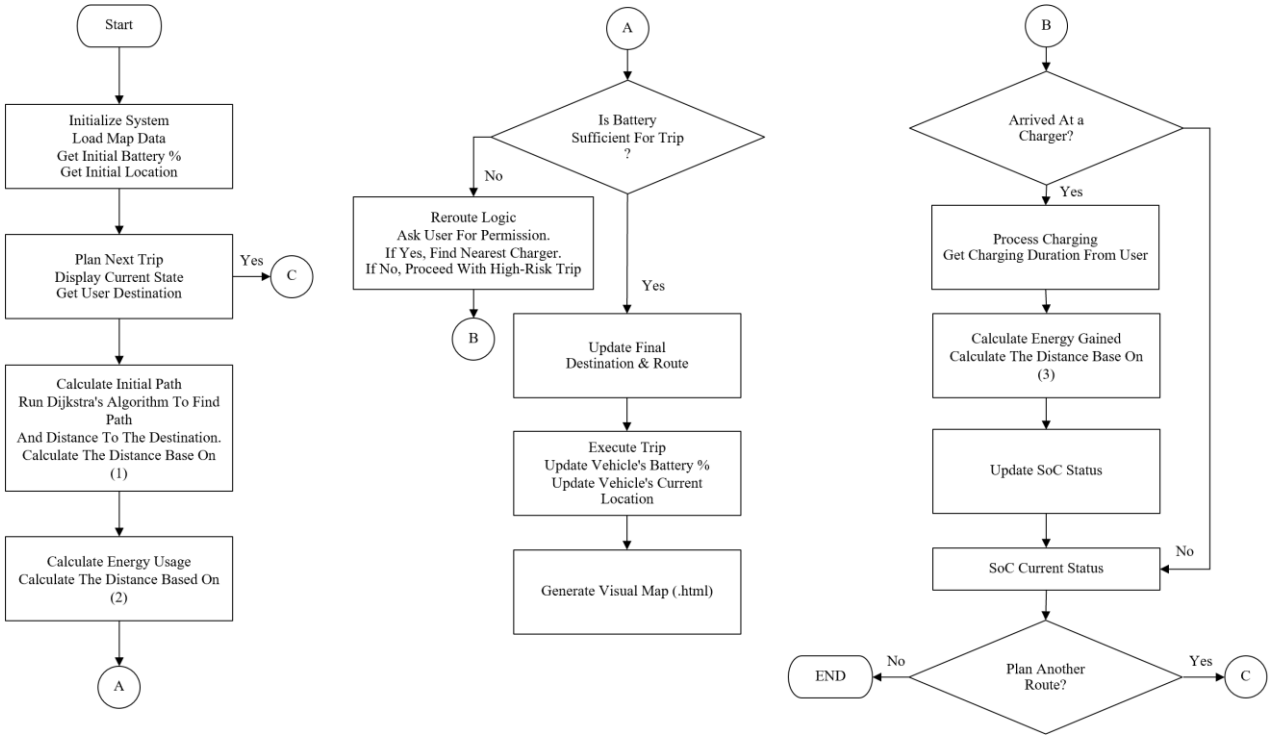

Fig. 3. System flowchart illustrating the process from initialisation to energy-aware routing, charging, and real-time navigation decisions.

2.3 System Flowchart of Dijkstra-Based Routing with Energy-Aware Logic

Figure. 3. shows the flowchart detailing the overall operational logic of the energy-aware route planning system implemented on Raspberry Pi 5, from initialisation to route execution and dynamic rerouting.

The proposed energy-aware navigation system's operation begins with the Initialisation Phase, in which a pre-processed road graph—generated from OSM data using the OSMnx library—is retrieved from local storage on the Raspberry Pi. Concurrently, the system acquires the vehicle's current SoC and GPS coordinates to establish the initial conditions.

Following this, the system proceeds to the Trip Planning stage, which displays the vehicle's current location and battery level to the user. The user then selects a desired destination through a graphical interface.

Once the destination is set, the system enters the path Calculation phase. Here, Dijkstra's algorithm is employed to compute the shortest path from the current location to the selected destination based on the locally stored weighted road graph, as described as shown in Eq. (1).

Additionally, the system estimates the total energy required for the trip using a predefined energy consumption rate derived from experimental data, as shown in Eq. (2).

This value is then compared with the current battery capacity. If the available energy is insufficient, the system initiates the rerouting Logic, where the user is prompted to confirm whether to redirect through the nearest charging station or proceed with the current route under energy risk.

Upon arrival at a charging station, the system requests the expected charging duration and updates the SoC accordingly. After charging, the system re-evaluates energy sufficiency and either resumes the original trip or loops back to recalculate a new route.

$$E_{trip} = e_{rate} \times d \tag{2}$$

where

E_{trip} = Total energy required for the trip

e_{rate} = Specific energy consumption rate per distance

d = Planned travel distance

Subsequently, the energy sufficiency check is performed by comparing the estimated energy demand with the available battery level. If sufficient energy is available, the system proceeds directly to the trip. However, if the battery is deemed insufficient, the system prompts the user with the option to reroute.

If the user accepts, the system locates the nearest charging station and updates the route accordingly; otherwise, it continues with the original high-risk path. Once routing is confirmed, the trip execution phase begins, involving real-time tracking of the vehicle's position and energy consumption, while simultaneously displaying the planned route via an interactive web-based map. If the updated path includes a charging station, the charging station handling process is activated upon arrival. After charging, the system enters the post-charging or trip completion phase to reassess the battery

status. If the updated SoC remains insufficient to reach the destination, the system halts and prompts the user to initiate a new route.

If accepted, the process loops back to the trip planning phase; the session is terminated if declined.

3. Implementation

3.1 Experimental Platform

To evaluate the performance of the proposed energy-aware navigation system, a ground-based wheeled drone was developed and employed as a platform for field testing, as shown in Fig. 4.

Fig. 4. Ground-based wheeled drone used to record energy consumption during route execution.

The objective of the test was to evaluate the system's ability to monitor and estimate energy consumption during actual route execution in a real-world environment.

To simulate realistic travel conditions, a ground-based wheeled drone was used to traverse the computed path generated by the navigation system physically. The test aimed to reflect consistent movement behavior similar to practical deployments. Throughout the experiment, the vehicle followed the planned route without manual intervention.

The system continuously tracked battery voltage and usage in real time, allowing for precise measurement of energy consumed during each trip. This empirical data provided a reliable basis for validating the energy estimation model integrated within the route planning algorithm. Moreover, conducting the test under controlled yet repeatable conditions ensured consistency across multiple runs, making it suitable for performance benchmarking. The resulting energy profiles contributed directly to assessing the algorithm's energy efficiency.

Fig. 5. shows the battery voltage monitoring circuit, which illustrates the electrical schematic used to monitor the battery voltage in real time. This hardware component was critical in ensuring accurate SoC readings and supporting the system's energy-aware decision-making process.

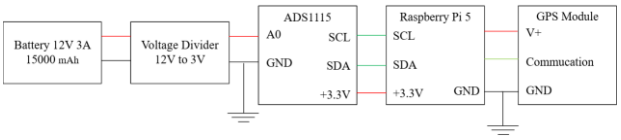

Fig. 5. Circuit diagram for battery voltage and GPS data acquisition using a voltage divider and ADS1115 connected to Raspberry Pi 5.

The system uses a 12V, 15,000 mAh lithium-ion battery connected to a resistive voltage divider circuit that scales the voltage down to approximately 3V—within a safe range for analogue input. The scaled voltage is fed into an ADS1115 Analog-to-Digital Converter (ADC), a high-resolution 16-bit device communicating with the Raspberry Pi 5 via the I2C protocol. This setup enables accurate and continuous battery voltage monitoring under varying load conditions.

The updated circuit diagram Fig. 5. combines voltage sensing and GPS acquisition into a unified hardware layout. The voltage divider reduces the 12V battery output to a safe level for the ADC input, and the ADS1115 digitizes the voltage for real-time processing by the Raspberry Pi. A GPS module is also integrated into the circuit, powered by the same 12V source and connected to the Raspberry Pi 5 through a dedicated communication line (e.g., UART). the GPS module provides real-time location data, essential for dynamic route planning and distance estimation.

This configuration ensures electrical safety and seamless location and energy data integration, enabling the system to make informed decisions. The measured battery voltage is converted to SoC using a predefined voltage-to-SoC mapping. This SoC value and GPS coordinates form the core input for the energy-aware routing algorithm. If the energy is insufficient to complete the planned trip, the system initiates rerouting to the nearest charging station, Including the GPS module enhances the system's ability to provide location-aware, energy-efficient navigation under real-world conditions.

4. Results and Discussion

This section presents the experimental results obtained from the developed vehicle platform. We first validate the performance of the real-time battery monitoring system and then analyze the vehicle's energy consumption characteristics. Finally, we discuss the implications of these findings for the primary goal of this work, enabling energy-aware pathfinding.

4.1 Discussion and Implications for Energy-Aware Pathfinding

Integrating the validated battery voltage sensing circuit with the empirically measured energy consumption profile establishes a robust and practical platform for developing and evaluating energy-aware navigation algorithms. The proposed system highlights a closed-loop relationship between real-time energy monitoring and

dynamic route planning, forming this research's core focus.

All computations, including graph processing, energy estimation, and pathfinding via Dijkstra's algorithm, are performed locally on a Raspberry Pi 5. This local processing approach reduces system latency, eliminates dependence on cloud computation, and ensures suitability for real-time operation in embedded environments.

Two test scenarios were designed under controlled conditions to assess system performance. In the first scenario as shown in Fig. 6, the vehicle traveled from the campus main gate to a predefined building. While the vehicle reached the destination, the battery's SoC dropped to triggering a low battery warning. This emphasizes the need for accurate energy prediction to ensure route feasibility. The system recorded a specific energy consumption rate of 3.45 Wh/km, a benchmark for estimating energy requirements in subsequent routes.

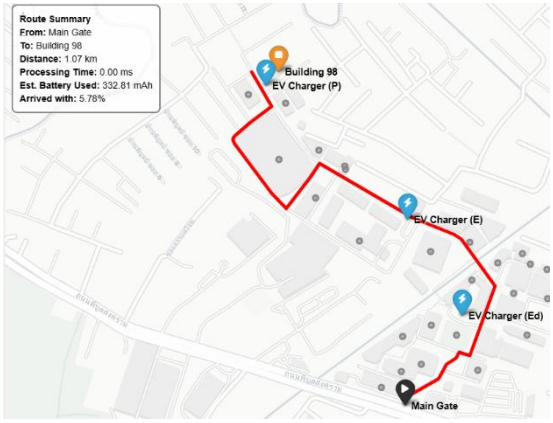

Fig. 6. Route followed without rerouting. The vehicle arrives with a critical battery level (5.78%), triggering a low battery warning.

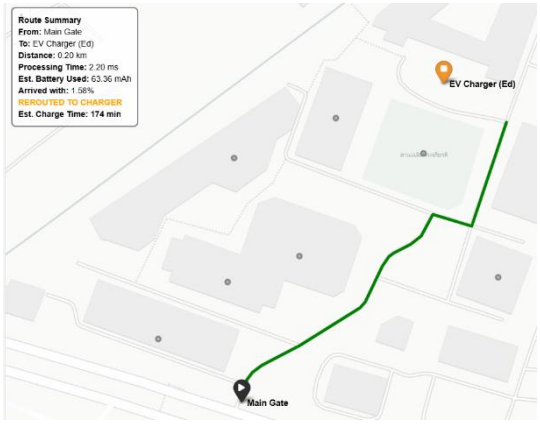

Fig. 7. Energy-aware rerouting to the nearest charging station due to insufficient battery for the original route.

In the second scenario Fig. 7, the system identified that the remaining energy was insufficient for the original route. As a result, the rerouting logic was triggered, redirecting the vehicle to the nearest charging station, the vehicle completed the revised route with only SoC remaining and initiated a charging procedure.

Based on real-time SoC feedback, this proactive adjustment validates the system's ability to make intelligent, energy-aware decisions and adaptively maintain safe navigation even under constrained energy conditions.

4.2 Computational Demands of Dijkstra Route Planning

To comprehensively evaluate the operational performance of electric vehicles, it is essential to assess energy consumption and computational processing time. While battery capacity and range are critical indicators of system longevity, the efficiency of the power system—especially during real-time operations—directly affects practical usability and sustainability. Simultaneously, computational demands from onboard systems, particularly for dynamic route planning, significantly influence responsiveness, processing overhead, and overall energy utilisation. This study investigates these two interconnected aspects by rigorously quantifying energy consumption and analyzing processing time under distinct routing scenarios. The goal is to understand how energy and processing constraints affect navigation performance and decision-making on embedded platforms.

An experimental evaluation was conducted using a 10 kg ground-based vehicle powered by an 11.1 V, 15 Ah lithium-ion (Li-ion) battery. During testing, the vehicle maintained a constant speed of 30 km/h for 30 minutes, covering 15 kilometers. Throughout the experiment, the system drew an average current of 9.33 A, leading to an average power consumption of 103.56 W. The total energy consumed was 51.78 Wh, with a corresponding battery drain rate of 155.5 mAh/min. To enable consistent comparisons across use cases, this value was normalized to a specific energy consumption rate of 3.45 Wh/km, which serves as a benchmark for evaluating the energy cost of travel in various path scenarios.

Table 1 summarizes the processing time for the standard route scenario. Across five test cycles, distances ranged from 0.81614 km to 1.07014 km, with processing times between 0.29 ms and 0.62 ms. The average processing time was 0.446 ms, and the average processing time per kilometer was calculated as 0.499 ms/km.

Table 2 shows the rerouting scenario, where distances ranged from 0.04452 km to 0.20372 km. Processing times ranged from 0.34 ms to 0.73 ms, with an average of 0.414 ms. The average processing time per kilometer slightly increased to 0.503 ms/km.

While the raw average processing time remained similar across both scenarios, a deeper comparison of processing time per meter reveals a significant disparity. The system used only 0.000499 ms/m for standard routing, while energy-constrained rerouting required 0.00503 ms/m—a computational cost approximately 10 times higher per meter. This highlights the added complexity of decision-making under battery constraints.

Table 1. Computation time of Dijkstra's algorithm for a direct route when battery capacity is sufficient

No. of Conducted Test Cycles	Distance (m)	Processing Time (ms)
1	0.86056 km	0.62 ms
2	0.81614 km	0.56 ms
3	0.86066 km	0.46 ms
4	1.07014 km	0.30 ms
5	0.86066 km	0.29 ms
Avg. Processing Time	0.446 ms	
Avg. Time per Distance	0.000499 ms/km	

Table 2. Computation time of Dijkstra's algorithm including rerouting logic when battery capacity is insufficient.

No. of Conducted Test Cycles	Distance (m)	Processing Time (ms)
1	0.04452 km	0.37 ms
2	0.06336 km	0.42 ms
3	0.04452 km	0.34 ms
4	0.05517 km	0.34 ms
5	0.20372 km	0.73 ms
Avg. Processing Time	0.414 ms	
Avg. Time per Distance	0.000503 ms/km	

The increased load results from additional computations required to assess battery levels, identify nearby charging stations, and determine whether alternate paths are feasible. These decisions must be made in real time, adding strain to the processing unit.

This tenfold increase in computational effort emphasizes the importance of efficient algorithms and local processing capability. Despite the higher complexity, the system—running entirely on a Raspberry Pi 5—maintained sub-millisecond processing times, validating its suitability for real-time, embedded applications.

The findings confirm that energy-aware navigation using Dijkstra's algorithm can be performed effectively on low-cost hardware, The normalised energy consumption value of 3.45 Wh/km and the consistent performance across routing tasks provide reliable metrics for future optimisation. Moreover, the insights from processing time analysis reinforce the importance of balancing algorithmic complexity with real-time performance when designing intelligent navigation systems for energy-constrained environments.

5. Conclusion

This study presents the implementation of an energy-aware navigation system on a Raspberry Pi 5 platform using Dijkstra's algorithm. The system uses a pre-processed road graph from OSM data and performs all routing computations locally, eliminating reliance on cloud-based processing. It integrates real-time GPS and battery SoC data to compute energy-efficient routes dynamically.

Dijkstra's algorithm calculates the shortest path based on distance and estimated energy usage. The system reroutes to the nearest charging station if the remaining battery is insufficient. A key advantage of this approach is its ability to run entirely on embedded hardware. Even when rerouting is triggered, computation times remain low—within the sub-millisecond range—demonstrating the feasibility of real-time path planning on compact, low-cost devices.

Validation was carried out using a ground-based wheeled drone under controlled conditions. Results confirmed that the system could compute feasible routes and estimate energy requirements accurately, achieving a consistent average consumption rate of 3.45 Wh/km. This value and live SoC data allow for intelligent navigation decisions aligned with current energy availability.

In conclusion, the proposed system confirms the effectiveness of Dijkstra's algorithm for energy-aware navigation and shows that local computation on embedded platforms can support real-time decision-making. This approach offers substantial potential for navigation in energy—and infrastructure-limited scenarios. Future work will explore more dynamic energy models, incorporate terrain data and acceleration profiles to improve estimation accuracy, and expand the system to support real-time obstacle detection using LiDAR.

References

[1] A. Abubakar, M. A. P. Mahmud, T. Khandoker, and A. L. Kiprakis, "An Energy-Aware Routing Algorithm for Solar-Powered Unmanned Aerial Vehicles," *2023 IEEE Global Communications Conference (GLOBECOM)*, Kuala Lumpur, Malaysia, 2023, pp. 4905–4910,

[2] J. Jeong, B. Ghaddar, N. Zufferey, and J. Nathwani, "Adaptive robust electric vehicle routing under energy consumption uncertainty," *Proceedings of the 2022 IEEE International Conference on Services Operations & Logistics, and Informatics (SOLI)*, pp. 1–6, Jul. 2022.

[3] S. Zhang, "An energy consumption model for electrical vehicle networks via extended federated-learning," *IEEE Access*, vol. 9, pp. 150,123–150,134, Nov. 2021.

[4] M. A. Navarro, L. Naranjo, and M. Á. Sotelo, "OpenStreetMap-based autonomous navigation with LiDAR naive-valley-path method," *IEEE Sensors Journal*, vol. 23, no. 2, pp. 1027–1037, Jan. 2023.

[5] Y. Modì, J. Bhattacharya, and P. Basak, "Estimation of energy consumption of electric vehicles using deep convolutional neural network to reduce driver's range anxiety," *ISA Transactions*, vol. 98, pp. 454–466, Mar. 2020. *(Indexed by IEEE)*