energies

MDPI

Article

Energy Consumption Analysis of the Selected Navigation Algorithms for Wheeled Mobile Robots

Adam Rapalski * and Sebastian Dudzik

Faculty of Electrical Engineering, Częstochowa University of Technology, 17 Armii Krajowej Avenue, 42-201 Częstochowa, Poland; sebastian.dudzik@pcz.pl

* Correspondence: adam.rapalski@pcz.pl

Abstract: The article presents the research on navigation algorithms of a wheeled mobile robot with the use of a vision mapping system and the analysis of energy consumption of selected navigation algorithms, such as RRT and A-star. Obstacle maps were made with the use of an RGBW camera, and binary occupation maps were also made, which were used to determine the traffic path. To recreate the routes in hardware, a programmed Pure Pursuit controller was used. The results of navigation were compared on the basis of the forward kinematics model and odometry measurements. Quantities such as current, except (x, y, phi), and linear and angular velocities were measured in real time. As a result of the conducted research, it was found that the RRT star algorithm consumes the least energy to reach the designated target in the designated environment.

Keywords: wheeled mobile robot; robotic navigation; energy consumption analysis; mapping; energy

Citation: Rapalski, A.; Dudzik, S. Energy Consumption Analysis of the Selected Navigation Algorithms for Wheeled Mobile Robots. *Energies* **2023**, *16*, 1532. https://doi.org/10.3390/en16031532

Academic Editor: Chunhua Liu

Received: 15 December 2022
Revised: 17 January 2023
Accepted: 1 February 2023
Published: 3 February 2023

1. Introduction

1.1. Navigation Algorithms of Mobile Robots

There has been significant progress in the development of technologies associated with the control of mobile robots. Dozens of new technological solutions are being developed to increase industrial productivity or support human work [1]. The development of mobile robots, with increasing ability to move in land, air or water [2,3], is considered to be a very fast-growing field of robotics. From the point of view of the mobility system used, we distinguish, among others, walking and wheeled robots. Mobile robots on wheels are widely used by people, from simple household activities such as cleaning (autonomous vacuum cleaners) to specialized works in contaminated environments, e.g., robots operating at the Fukushima nuclear power site after the tsunami [4,5].

Differential drive kinematics is a mathematical relationship that maps independent wheel motion to resultant motion of the robot chassis. This mapping is a fundamental issue for controlling all mobile robots, and it is mainly responsible for the predictable mobility of the robot. These types of mobile robots move around a circle with a specific radius, thanks to the different rotational speeds of the left wheel v_L and the right wheel v_R. Thus, assuming that there is no wheel slip, the robot platform can move along a horizontal plane on a straight or curved trajectory, and it can also rotate in place at a speed of v_C. Figure 1 shows the wheel radius as r, as well as the rotational speeds of the left ω_L and right ω_R wheels [6]. In the mathematical mapping of the drive kinematics, we assume that the wheels are not subject to skidding, and the movement of the wheels is limited to the movement along their directions forward and backward. This, together with the inherent limitation imposed by the robot on the chassis connecting the two wheels together, means that all rotations of the robot chassis have to be close to the point which lies along the common axle of the wheels. If both wheels rotate at the same speed, the robot rotates around a point substantially far from the robot with radius length r_{ICC} [7]. The relationships are illustrated in the methodology section in Equations (7) and (8).

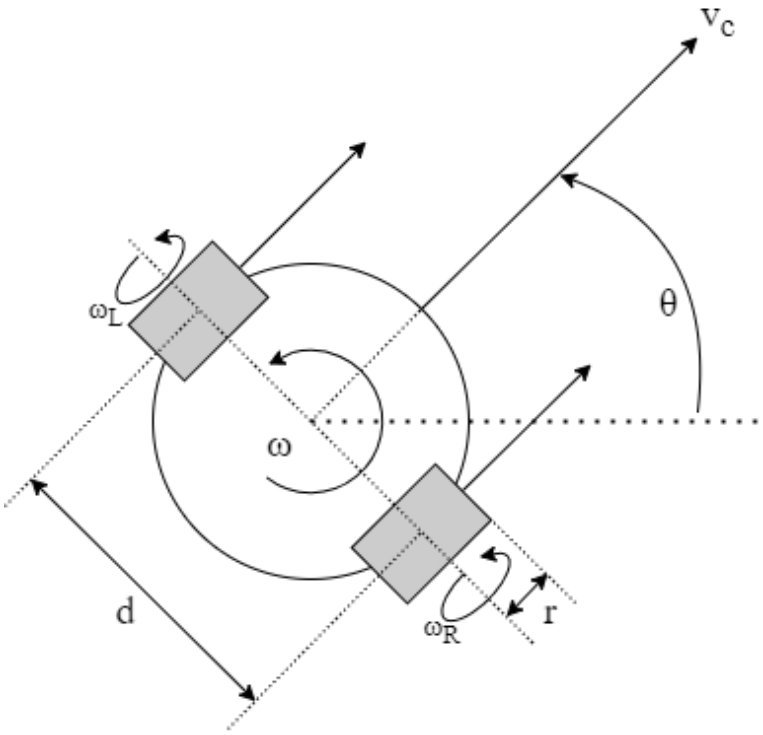

Figure 1. Geometric representation of kinematics of the differential drive.

In robotics of wheeled mobile robots, four-wheeled robots are often used. Models of the kinematics of individual types of drive are presented below according to the number of driven wheels and number of steering axles [8,9].

Four wheels with car kinematics: Reflecting the Ackermann steering geometry system, this vehicle model has four wheels separated by a distance, but only the front axle is steerable. The vehicle position is in the middle of the rear axle. The wheels on the drive axle can rotate in both directions. The required inputs for the kinematics model are front wheel speed ω_f (rad/s) and rear wheels ω_r (rad/s) and the steering angle of the rear wheels θ_r (rad). On the output of the model, we obtain data on the linear velocity v_x (m/s) and v_y (m/s) and the angular velocity ω (rad/s) [10].

Four-wheel steering kinematics: Such a vehicle has four wheels, which can be driven and steered independently, and it has two steering axles on the front and rear wheels. For the sake of simplicity, we assume the Ackermann control in such a way that the vehicle can be approximated as a two-wheeled system (colloquially referred to as a bicycle model). The required inputs for the kinematics model are front wheel speed ω_f (rad/s) and rear wheel speed ω_r (rad/s) and the steering angle of the front θ_f (rad) and rear θ_r (rad) wheels. On the output of the model, we obtain data on the linear velocity v_x (m/s) and v_y (m/s) and the angular velocity ω (rad/s) [8,10].

Four-wheel mecanum kinematics: In this case, the vehicle has four mecanum wheels that can be driven independently; there are no steering axles as there are four inputs and 3 degrees of freedom, so the system is distorted. The required inputs for the kinematics model are the speed of each of the wheels ω_1 (rad/s), ω_2 (rad/s), ω_3 (rad/s) and ω_4 (rad/s). At the output of the model, we obtain data on the linear velocity v_x (m/s) and v_y (m/s) and the angular velocity ω (rad/s) [8,10].

Each of the newer solutions is being developed in order to improve communication between robots and their autonomous capabilities. Robots are equipped with tools required to perform the tasks they are faced with. Realization is supported by various locators such as GPS, video systems and other additional sensors. In order to obtain smooth mobility within the environment, it is necessary for the robot to be able to map its surroundings and then to operate in them. Navigation algorithms, the purpose of which is to find a collision-free

path of movement of the robot from the initial position, along with orientation, to the target position, are of utmost importance [11–13].

Although path-planning algorithms differ in detail, most of them share common assumptions. The first step is to know the geometry and kinematics of a mobile robot that can have a complex geometric shape and then to map the path to a point in space, which is often called configuration space [14]. This mapping transforms the original issue into a path planning issue for a moving point. Next, we digitize the configuration space and construct a chart that represents the connectivity of the space. Finally, we review this chart in order to map the path to the target [15,16].

The path-mapping problem is essentially a question of connecting the start and end points. To answer this question, the path-planning algorithm creates a chart and then reviews it to find the right route. The first step—so-called graph construction—is crucial. This is what makes the path-planning algorithms different from each other. There are three general approaches to path planning present in the literature: roadmap, cell decomposition and potential field [15].

The robot Quanser QBot 2e on the top comes with a Microsoft Kinect sensor that has the ability to generate a depth map of the environment. Regarding the location and orientation of the robot chassis, this information can be helpful for autonomous map building. Sensor Kinect has the following limitation of parameters:

- Range: between 0.5 and 5 m. If an object is outside these values, the data are invalid and will be zeroed down by the software.
- Field of View: The Kinect sensor has a horizontal field of view limited to 57°. The robot should be rotated as well to map the entire 360°.

A 480 × 640 depth image is a representation of Kinect sensor depth data in QUARC. For simplicity, there is one row of the depth image data for 2D mapping—for each row, only 640 pixels will be used. Distance to the object (in millimeters) is calculated on the basis of the RGBD camera measurements.

Navigating the terrain and finding the shortest path to the destination location is one of the primary problems when mapping a path. Although there are many approaches to this problem, one of the most popular and well-known is the A-star search algorithm, as it is quite simple and flexible. The A-star algorithm is a heuristic algorithm designed for finding the shortest path in a graph. This is a complete and optimal algorithm, which means always finding the best solution. This method includes an organized memory model that guarantees that each point can be visited. A-star is an example of the "best first" method. The algorithm works best when the search space is a wooded area. Grid-based algorithms assume that the field is divided into a finite number of cells, and then we apply the A-star path-planning algorithm. First, samples are taken from space to create a discrete C space. Note that this algorithm requires all marked vertices to be stored. The amount of data stored (strongly) depends on the dimension of the configuration space and (linearly) depends on the resolution, so this algorithm is not practical for multidimensional C space [17].

It can be deducted according to the given information about the target note, the current node, and the obstacle nodes how to find the best next node that can be added it to the list. A-star uses a heuristic algorithm to drive the search, while calculating the path at minimal cost, it also calculates the distance (also known as cost) between the current location in the workspace or current node and the destination. It then evaluates all adjacent nodes that are open (i.e., not an obstacle or have not already been visited) for the distance that is expected or a heuristic estimated cost from them to the target, which is called heuristic cost, $h(n)$. It specifies the cost of moving from the current node to the next node, which is called the path cost $g(n)$. Thus, the total cost of reaching the destination node, as shown in Equation (1), is calculated for each subsequent node, and the node with the smallest cost is selected as the next point.

$$f(n) = h(n) + g(n), \tag{1}$$

The operation of the algorithm is based on the minimization of the objective function $f(x)$, which is defined as the sum of the cost function $g(x)$ and the heuristic function $h(x)$.

In each step, the A-star algorithm extends the already created path by another vertex of the graph, selecting the one in which the function value will be the smallest. The function $g(x)$ determines the real cost of reaching point x (the sum of the edge weights that already belong to the path and the weight of the edge connecting the current node with x node). The function $f(h)$ is called a heuristic function. It estimates (always optimistically) the cost of moving from point x to the target vertex. A valid heuristic function must meet two conditions:

- Acceptability condition: $g(x) + h(x) \leq g(x_t)$;
- Monotony condition: $g(x_j) + h(x_j) \geq g(x_i) + h(x_i)$, where the end condition $j > i$, and x_i is the endpoint.

Rapidly exploring random tree (RRT) is an algorithm designed to efficiently search two-dimensional as well as multidimensional spaces by randomly building a tree-filling the space between map boundaries or obstacles [18]. The tree is constructed incrementally from samples randomly selected from the search space, and it is inherently geared to grow toward large, untrawled areas of the map. The RRT algorithm easily deals with obstacles and differential constraints problems, including nonholonomic and kinodynamic constraints. RRT algorithms and its variations are widely used in planning the movement of autonomous mobile robots. The algorithm can be seen as an open-loop trajectory generation technique for nonlinear systems with state constraints. RRT can also be considered as the Monte-Carlo method to search for deviations in the largest regions of the Voronoi graph in the configuration space. Some variations of the RRT algorithm draw heavily on the effects of stochastic fractals [19,20].

Fast Exploring Random Tree (RRT) is an algorithm designed to efficiently search nonconvex, multidimensional spaces by randomly building a space-filling tree. The tree is constructed incrementally from samples taken at random from the search space and is inherently designed to grow towards large, unexplored problem areas. RRT was developed by Steven M. LaValle and James J. Kuffner Jr. They easily deal with obstacle problems and differential constraints (non-holonomic and kinodynamic) and are widely used in autonomous robotic motion planning.

RRT grows a tree rooted in the initial configuration using random samples from the search space. As each sample is drawn, a connection is attempted between it and the nearest state in the tree. If the join is viable (passes completely through free space and respects all constraints), this results in a new state being added to the tree. With uniform sampling of the search space, the probability of expanding an existing area is proportional to the size of its Voronoi region. Since the largest Voronoi regions belong to the search frontier, this means that the tree preferentially expands toward large, unexplored areas.

The length of the connection between the tree and the new state is often limited by the growth factor. If the random sample is farther from the nearest state in the tree than the constraint allows (e.g., the presence of an obstacle), the new state at the maximum distance from the tree along the line to the random sample is used instead of the random sample itself. Random samples can thus be seen as controlling the tree's growth direction, while the growth factor determines its rate. This maintains the tendency to fill the RRT space while limiting the size of the incremental growth [21,22].

The properties of RRT-based algorithms have benefited many applications. Virtual reality planning uses the RRT as high-degree virtual humans and in the artificial intelligence on game development, the RRT is used as a tool to plan random paths quickly [2] and exploring levels. In robotics, a path can be created for systems such as manipulators,

mobile robots, underwater robots, helicopters, humanoids and space robots. Path planners are associated to many types of autonomous vehicles that were also created over the RRT structure, such as Aerial Vehicles (Micro-Vehicle Aircraft, Hypersonic Air Vehicles, Blastless Unmanned Aerial Vehicles, Unmanned Aerial Vehicles and Combat Unmanned Aerial Vehicles), Unmanned Ground Vehicles (Unmanned Terrestrial Vehicles) as well as Unmanned Surface Vehicles (Autonomous Submarine Vehicles, Unmanned Maritime Surface Vehicles) [1,2,6].

The RRT algorithm during operation develops the rooted tree from the initial configuration, generating initially random points (samples) on the entire surface of the map or in three-dimensional space. When determining each sample, an attempt is made to connect it to the closest state in the tree. If the connection is feasible (it goes completely through the free space and meets all constraints), it results in adding a new state (branch) to the tree. With uniform sampling of the search space, the probability of an existing state's expansion is proportional to the size of its region. Since the largest regions of the Voronoi graph belong to the states on the boundary of the search, this means that the tree in the preferential form expands where there are large, unexplored areas of the terrain [21,23]. This mechanism is shown in Figure 2.

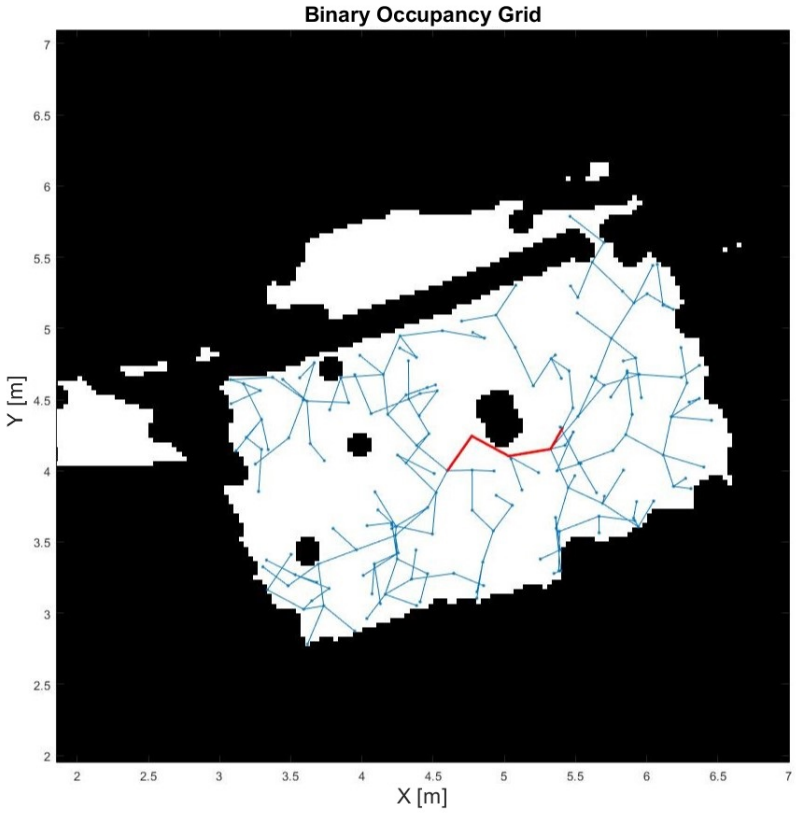

Figure 2. The generated tree and the found path from the start point to the end point.

The increase in RRT can be biased by increasing the sampling probability of states in a certain area. Most of the practical solutions of RRT in robotics use this mechanism to guide the search toward a target point in a path. This is completed by introducing the sampling probability to the designated target, the so-called occupancy grid. The higher the probability, the more likely the tree will grow toward its destination [24].

RRT star is the optimized version of the RRT algorithm. As the number of nodes approaches infinity, the RRT star algorithm will provide the shortest possible path to the goal. While this is realistically not feasible, this statement suggests that the algorithm works to come up with the shortest path. The basic principle of RRT star is the same as RRT, but the two key additions to the algorithm yield significantly different results. The first one

is that the RRT star registers the distance traveled by each vertex in relation to the parent vertex, the so-called tree branch/top cost. The second difference is that when a new state of the tree is joined with the cheapest neighbor, the neighbors are checked again. The neighbors are checked to see if re-wiring to the newly added vertex will reduce their cost of reaching their goal. This feature makes the path smoother. RRT star creates incredibly straightforward paths. Additionally, its graphs are characteristically different from the RRT graphs.

Another method of determining the path from the start point to the end point, while maintaining the distance from obstacles, is the potential field method [25]. In this approach, the target shows an impressive potential field (U_{target}) and the obstacles exhibit repulsive potential fields (U_{obsi}) [26]. The resulting potential field arises by summing attractive and repulsive potential fields as follows:

$$U = U_{target} + \sum U_{obsi}, \tag{2}$$

The net potential field F is used to produce an artificial force equivalent to the gradient of the potential field with a negative sign, as follows:

$$F = -\nabla U, \tag{3}$$

The movement is then conducted in the direction of the generated force. The target field and obstacles can be defined based on the distance to the target and the obstructions.

$$U_{target} = \frac{dist_{tar}(x,y)}{dist_{tar,th}} - 1,$$
$$U_{obsi} = \begin{cases} 0 & if \quad dist_{obsi} > dis_{obsi,th} \\ 1 - \frac{dist_{obsi}(x,y)}{dis_{obsi,th}} & otherwise \end{cases} \tag{4}$$

where $dist_{tar}(x,y)$ is the distance to the target (x, y) location of the working area, $dist_{obsi}$ is the distance to the i^{th} th obstacle point from the (x, y) position of the workspace, $dist_{tar,th}$ is a normalization factor set to the diagonal distance of the workspace, and $dis_{obsi,th}$ is the radial boundary of the obstacle potential field.

The Pure Pursuit algorithm allows the robot to follow a route that is designated. Calculating the curvature of the vehicle route is the way to track it each time. It estimates the angular speed that moves the robot from its current position to reach the pre-emptive point, which is located in front of the robot. It is expected that the linear speed is constant, so it can be altered at any point. In that case, the algorithm moves the pre-emptive point on the path based on the robot's current position to the last point of the path. It can be observed as a robot that is constantly chasing a point in front of it. As a result of the algorithm, the robot moves in pursuit of a momentary target, which is located along the path. According to that, the vehicle recreates the given path with some precision. In the case of differential drive kinematics, these are pre-set linear parameters: speed and maximum angular velocity. An important issue is the choice of the parameter of distance from the point in front of the robot related to the implementation of the described method. The tracking accuracy decreases if the forward-looking distance is too large. On the other hand, the movement of the robot displays oscillations around the set path if the forward-looking distance is too small [7].

It can be easily noticed that propulsion and steering are geometrically decoupled if the vehicle's coordinate system is placed at the rear differential with the x-axis colinear to the rear axle. It is important to remember that (x, y) is constrained to be on the path. The objective is to calculate the curvature of the arc that joins the origin to (x, y) and whose chord length is l [7]. From Figure 3 and the geometric equations, it can be concluded:

$$x^2 + y^2 = l^2, \tag{5}$$

$$x + d = r \tag{6}$$

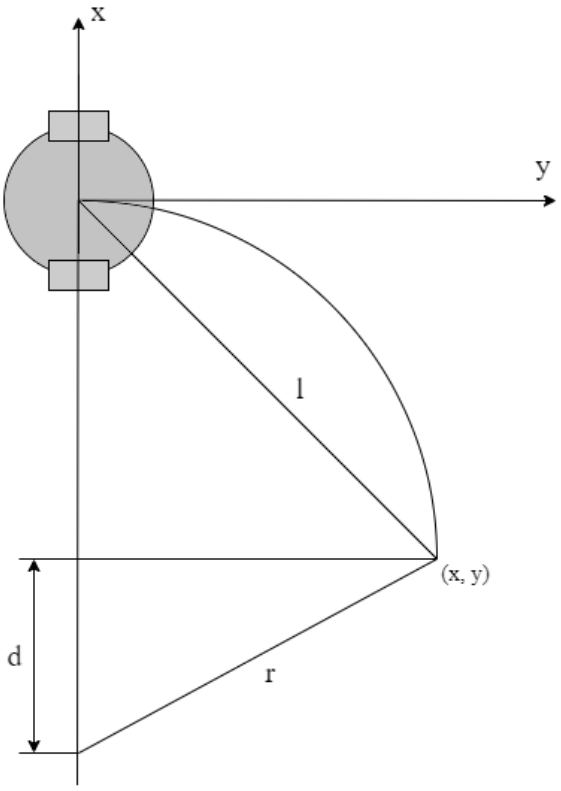

Figure 3. Geometric representation of the Pure Pursuit algorithm [7].

1.2. Robot Powering Systems

Mobile robots need an energy source to perform all kinds of tasks. Today, the most commonly used batteries are lithium-based, i.e., lithium-ion (Li-Ion) or lithium-polymer (Li-Po or Li-Poly). They have a higher energy density than other technologies, are slower to self-discharge and have no memory effect. They should not be overcharged or significantly discharged during use. Manufacturers use hardware or software protection on receivers or chargers to avoid these situations.

At the turn of the 20th century, emphasis was placed on creating and improving navigation algorithms that increase the energy efficiency of robot motion by optimizing, among other things, the speed of the unit. Nowadays, neural networks [27], modern algorithms (e.g., SLAM) and increasingly sophisticated sensors are used to minimize energy consumption by improving path planning and pre-computing robot movements in a given environment. Robots moving in the field have limited energy resources, most of which are transferred to motion systems. Power is also needed to operate system units, logic components or sensors installed on the robot [28,29].

The development of mobile robots is constrained by the limitations of batteries and their charging points. By optimally directing energy to individual components, wear and tear can be reduced, and the robot can operate for longer on a single charge. Navigation algorithms have been studied to minimize power consumption. In [30], methods for robot locomotion were investigated, taking into account, among other things, motor dynamics and a three-phase velocity profile. Savings of up to 30% in energy consumption have been observed at different levels. The following research areas are related to the dynamic evaluation of the efficiency of the battery capacity, the discharge rate or the energy needed to reach the destination. In [31], an energy management method is proposed to estimate the risk of battery depletion. Exceeding the risk parameter forces the robot to return to the charging station. During robot operation, a decision has to be made as to whether to redirect the robot to the charging station or to continue the task. The task level performed in relation

to the battery charge level is considered. In addition, the method can be used to assess whether the task is justified if the batteries do not have enough power to complete the task. An example is mobile cleaning robots (vacuuming, mopping), which need to consider the battery charge level and the remaining map to be cleaned during operation [32].

Researchers are currently working on the design of a robot trajectory that takes into account optimal energy consumption. In [33], authors Liu and Sun proposed a model for the energy consumption of a mobile robot, which was further validated in the laboratory. They used the developed model to design the motion of a mobile robot with minimum energy consumption. In subsequent articles [34], a new model was proposed using an A-approximation algorithm with an added energy-saving criterion. The time and speed to reach successive waypoints were optimized. The state-of-the-art technology is also based on an advanced battery management system (BMS). Algorithms are developed and verified under rigorous laboratory conditions. The paper [35] presents the idea of a BMS using a Kalman filter [36]. Energy saving is also essential in the design of multi-helicopter flight paths [23].

2. Materials and Methods

2.1. The Purpose of the Research

The aim of the study was to compare some selected navigation algorithms in terms of electricity consumption by the robot. For a given obstacle system, a motion path was determined using the selected navigation algorithms. Then, the given route was played in a hardware in the loop by the programmed Pure Pursuit controller. During the passage of the robot, energy consumption measurements were carried out by the robot's chassis (two DC motors). In addition, in real time, such quantities as the current beyond (x, y, phi) and linear and angular velocities were measured.

2.2. Setup for Evaluating the Navigation Algorithms for Mobile Wheeled Robots

The research will utilize a novel laboratory equipped with a hardware and software kit—the Autonomous Vehicles Research Studio from QUANSER—to conduct research related to wheeled mobile robots. At their disposal, there are two wheeled robots: QBot 2e. Additionally, there is a Ground Control Station and eight cameras (Flex 13) from Optitrack [6,37].

The control station is built on a PC with the MATLAB 2018a programming environment. Furthermore, Motive 2.0 and QUARC Real-Time Control Software 2018 are also installed on the control station. For communication purposes, a router with two frequencies of 2.4 Ghz and 5 Ghz is applied [6].

The QBot 2e mobile robot is one of the elements of the environment This is an autonomous ground robot with an open architecture built on a two-wheeled Kobuki mobile platform. QBot 2e utilizes a differential drive mechanism. It is a compound with two central driving wheels mounted on a common axis that bisects the robot. The front and back wheels of the robot stabilize the platform without compromising mobility. Both drive wheels can be driven forward and backward independently to trigger a movement different than the base of the robot. This approach to the wheel geometry of a mobile robot is very common due to its simplicity and maneuverability, and 23.5 cm is the distance between the left and right wheel. The vehicle diameter is 35 cm, and its height (without accessories) is 10 cm. It reaches 27 cm with the Kinect sensor installed. The platform can move at a maximum speed of 0.7 m/s. The robot uses a differential drive mechanism. The front and back wheels of the robot stabilize the platform without compromising mobility. Encoders are used to measure the movement of each wheel, and an integrated gyroscope (IMU) estimates the robot orientation or angle of deviation. QBot 2e has integrated impact sensors (left, right and center) and edge sensors (left, right and center). The robot is driven by a Raspberry Pi 3 B+ on-board computer and an integrated wireless LAN, which allows wireless connection between the test station and/or other vehicles. The robot has the Microsoft Kinect video system, which allows processing RGB and depth

data for various purposes, including visual inspection, 2D and 3D mesh mapping. The resolution of the cameras is 640 × 480 pixels. The Kinect depth sensor uses infrared light and has a range of 0.5 to 6 m. QBot 2e is powered by a lithium-ion battery supplied by Yujin Robot. The battery fits under QBot 2e and can run unceasingly for about 3 hours when it is fully charged. The total weight of the robot is 3.82 kg [6,38].

QBot 2e has a pair of coaxial wheels. These wheels are driven by high-performance DC motors with encoders and drop sensors. In order to determine the relationship between the independent motion of the wheels and the resultant motion of the robot, we start with modeling the motion of the robot about the center of the wheel axis. If the radius of the wheels is denoted by r, and the wheel rotational speed is denoted by ω_L and ω_R for the left and right wheel, respectively, the linear speed along a given path can be described as follows [6,37,38]:

$$\begin{aligned} v_L &= \omega_L\, r, \\ v_R &= \omega_R\, r, \end{aligned} \tag{7}$$

where v_L is the linear speed of the left wheel and v_R is the linear speed of the right wheel. Assuming there is no wheel slip, QBot 2e moves on the horizontal plane along a straight line or along a curved path. It can also spin on a spot by changing the relative speed between the left and right wheel [38].

The simple and inverse kinematics equations of the differential drive describe the movement of the QBot 2e wheeled mobile robot. Simple kinematics is used to locate the robot in the working space, while inverse kinematics allows one to control the robot and program its movement. The diagram of a mobile robot in a local reference frame, along with basic kinematics parameters, is shown in Figure 4.

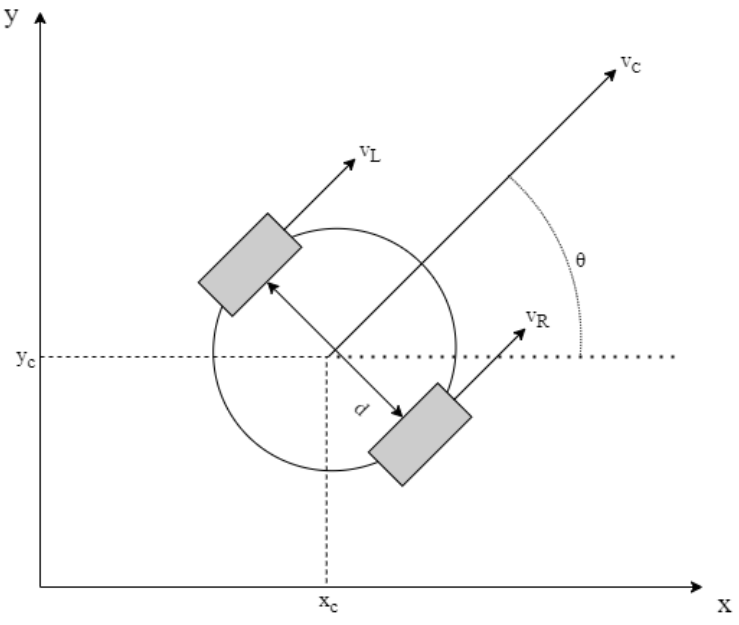

Figure 4. Diagram of the QBot 2e mobile robot.

Taking into account the markings, the following equations can be derived to describe the robot's movement:

$$\begin{aligned} v_c &= \frac{v_R + v_L}{2}, \\ \omega_c &= \dot{\theta} = \frac{v_R - v_L}{d}, \\ r_{ICC} &= d\frac{v_R + v_L}{2(v_R - v_L)}, \end{aligned} \tag{8}$$

where v_C is robot speed, ω_c is the angular velocity, θ is the heading angle of the robot and d is the distance between the left and right wheels.

The working space is surrounded by a square-shaped mesh, and the floor is lined with non-slip panels. The whole is 5×5 m in size, which is a fairly large area of possible trajectory of movement. The performance includes linear, circular and polygonal motion trajectories. In the corners of the cage and in the middle of the sides, there are eight Opti Flex cameras, which form the basis of the motion capture system [39].

The lab is equipped with MATLAB Simulink software and QUARC drivers from QUANSER. It enables the development of control algorithms for mobile robots. These algorithms can also be created from scratch using only blocks or fragments of the Simulink blocks provided by QUANSER [6,38,39].

Real-time code is generated directly from drivers designed by Simulink by QUARC Real-Time Control Software 2018 SP1 and are run in real time on the target Windows system [39].

Two models created in Simulink are often used due to the need for communication between the workstation and the robot. Mission Server that is one of them allows planning the route of the robot, and the other (Stabilizer) is responsible for its implementation. Both of the mentioned models are running on a different target hardware platform. Mission Server runs on a base station under Windows, and Stabilizer runs on the Linux operating system of the robot being tested. Models exchange data via TCP IP and a wireless router [39].

3. Methodology of Research

In the following section, the methodology of testing QBot 2e navigation algorithms is described. The stages of the research were as follows: The further part of this work includes the methodology of research of three navigation algorithms of QBot 2e.

1. Create a map using the Microsoft Kinect sensor, with the ability to generate a depth map of the environment, along with the location and orientation of the robot's chassis;
2. On the map, binarization was performed at level 101, an obstacle mesh was created, and so was a binary occupancy map object. A local coordinate system is declared;
3. The obstacles on the occupancy grid are thickened by 0.07 m;
4. Defining the start and end points of the path, values for individual maps are presented in Table 1;

Table 1. The table shows designated start and end points (x, y).

	Map 1	**Map 2**	**Map 3**
Starting point	(0, 0)	(0, 0)	(0, 0)
End point	(0.5, −3.3)	(−3.5, −3.8)	(3.5, −3.55)

5. Launch of planners to create a waypoint path along with orientation:
   (a) plannerRRT;
   (b) plannerRRTstar;
   (c) plannerHybridAstar;
6. Programming the robot's movement according to the assumed path in the Matlab/Simulink environment;

7. Compilation of the simulation model and downloading it to the platform running in the real-time target;
8. Running the hardware in the loop simulation (HIL) of the model;
9. Analysis of simulation results.

Before the experiment, the position of the RGBW camera, which is the Kinect, was checked, and it was set so that it was parallel to the floor plane and not tilted. The resulting vector is used in the QUARC models to match the depth data to real-world coordinates.

To map the surrounding space of the Qbot 2e robot, the Simulink model was used. The diagram is shown in Figure 5. The main element of the scheme is the 2D Mapping block, which generates the occupancy map required for later delineation of paths avoiding obstacles.

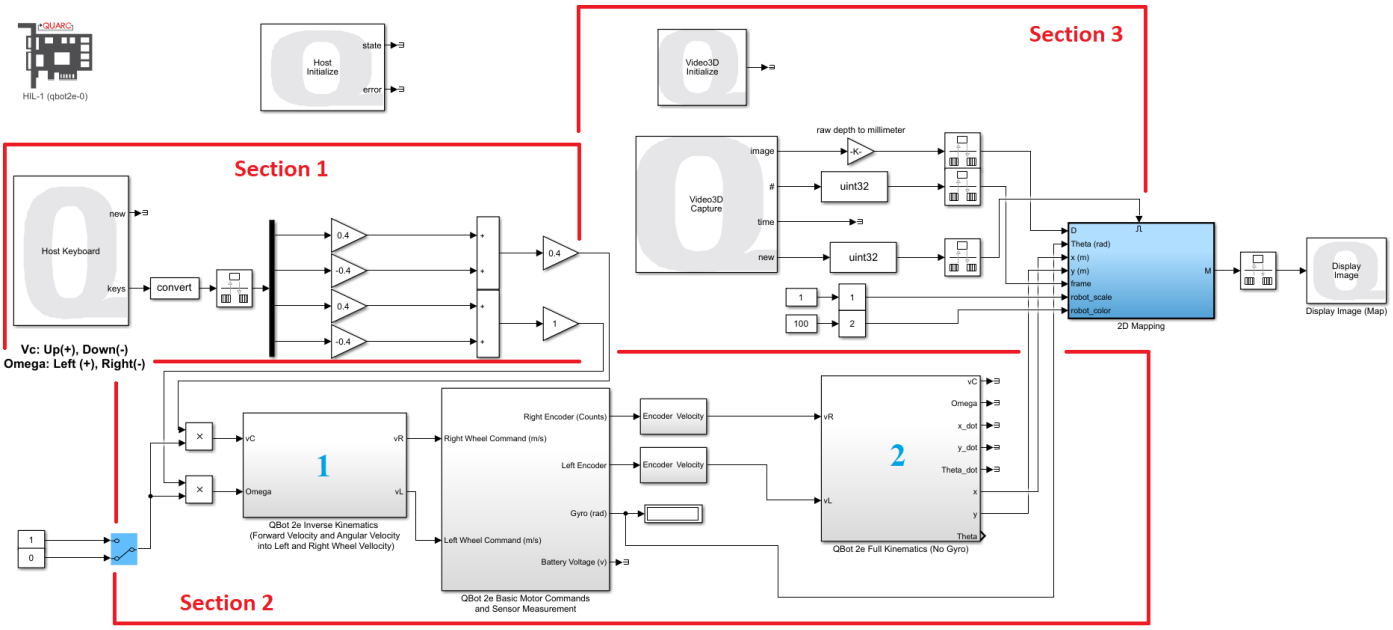

Figure 5. Simulink diagram for controller model for the mapping of QBot 2e.

Section 1 in Figure 5 is responsible for controlling the robot during mapping; the robot can move around the entire available space by scanning subsequent parts of the map, including the areas behind the next obstacles. The Host Keyboard block reads the state of specific virtual keys on the host and generates a vector of boolean values to indicate which keys were pressed at the current time of sampling. Of course, the block can read the status of more than one key at the same time. The keys to be monitored are selected in the Block Parameters dialog box, which displays the so-called key list for keyboard keys. The Host Initialize block is required for proper operation; it supports the use of the mouse and keyboard along with other peripheral devices in the Simulink diagram being built.

Section 2 in Figure 5 is responsible for compiling the keyboard signals for the speed of the left and right wheels in the QBot 2e Inverse Kinematics block marked in Figure 5 as number 1 and QBot 2e Basic Motor Commands. The QBot 2e Full Kinematics (No Gyro) block is marked in Figure 5 as number 2. It outputs the x and y values, i.e., the robot's current location on the map.

Section 3 in Figure 5 illustrates the results on the diagram of the graph displayed during operation, which can be seen in Figure 6a.

The Video3D Capture block captures video images from a 3D image processing device, such as an RGBD camera, and displays each frame as a color, grayscale, or depth image. An RGB image is a three-dimensional matrix that shows the red, green, and blue components of each pixel in the image. A grayscale image is a two-dimensional matrix representing the luminance of each pixel in the image. The Depth Image is a 2D matrix that

shows the distance in meters from the camera to each pixel in the image. The Video3D Initialize block is necessary for the configuration of the block described above. It configures the video device name, the source of the sensor data (device or file), and the hardware to use. The Video Display block shows the video in a window on the host. It is created for common video frame rates. When the movie is paused and resumed, the current frame can be saved as an image on the disc.

Figure 6a shows the generated occupancy grid as a MatLab data file. Then, binarization was performed at the level of 101, creating a map of occupation with obstacles. Localized walls along with a gray unmapped matrix space were assumed as an obstacle. A coordinate grid has been declared, which is shown in Figure 6b. Then, to avoid making the map more detailed, smaller obstacles and noise were grouped into more complex blocks. For this purpose, the inflate was thickened by 0.07 m. The final map for the robot motion trajectory planners is shown in Figure 6c.

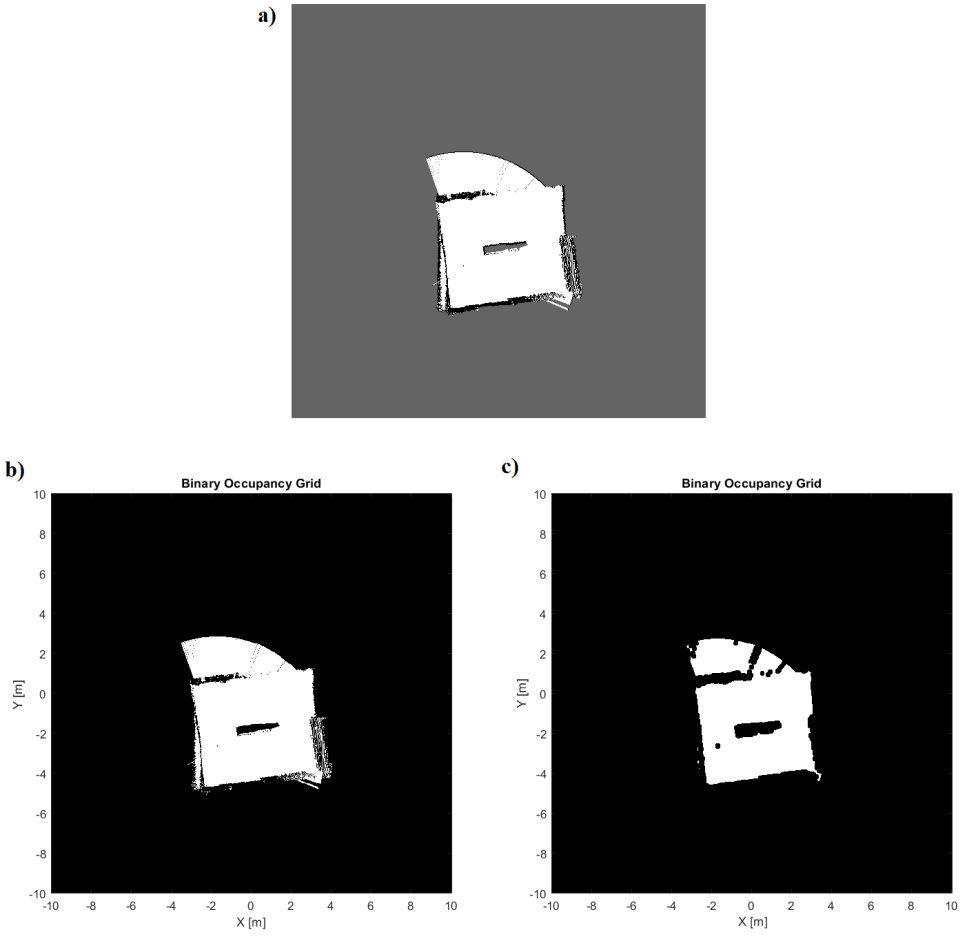

Figure 6. Stages of creating a grid of obstacles. In subfigure (**a**) shows the generated occupancy grid. In subfigure (**b**) shown is declared coordinate grid. In subfigure (**c**). shows robot motion trajectory planners.

Three generated maps with one obstacle, a map with two rooms and a narrow passage and a labyrinth were used for the study. Three path-planning algorithms were used on each of the maps, they were:

- The RRT algorithm planner, RRT
  The RRT planner was used to create an RRT random tree in order to find the geometric path to reach the designated destination on the map. The RRT is a tool which builds a random RRT tree on an occupancy map from samples randomly distributed on a ma-

trix. It is based on the incremental search. Ultimately, we obtain the searched space with the path start and destination combined. The great advantage of the algorithm is its speed of operation.
The same values of the RRT traffic planner have been set for each of the maps. The ValidationDistance setting was set to 0.01 m. The parameter of the maximum connection distance of points for subsequent nodes of the tree MaxConnectionDistance was set to 0.4 m. Each of the maps, however, had separate starting points and destinations of the paths, which are presented in Table 1.

- The RRT star algorithm planner, RRTstar
  The created planner RRTStar creates an asymptotically optimal RRT planner, i.e., RRT star. The RRT star algorithm additionally examines the vertices of the random tree in terms of the cost of getting to the destination (becoming closer to it). Thus, the algorithm copes better with more obstacles or narrow passages on the map.
  For each of the maps, the same values of the RRT star traffic planner were set. The ValidationDistance validation distance setting was set to 0.01 m. The parameter of the maximum connection distance of points for subsequent nodes of the tree MaxConnectionDistance was set to 0.4 m. For maps 1 and 2, the maximum number of iterations, creating nodes, was set at 2500, due to the complexity of map 3 being a maze and requiring further study of the occupancy grid, the parameter MaxIterations was changed to 7500. As before, each of the maps also had separate starting points and destinations of the paths, presented in the Table 1.
- The HybridA star algorithm planner, HybridAstar
  The A-star hybrid path planner generates a smooth traffic path in the 2D occupancy map created. It is used in the case of robots with nonholonomic constraints.
  For maps 1 and 2, the same values of the traffic planner were set. The ValidationDistance setting was set to 0.1 m. The MinTurningRadius and MotionPrimitiveLength values were set at 1 m. For the complex maze in map 3, motion planner values were specified: ValidationDistance 0.05 m, MinTurningRadius and MotionPrimitiveLength 1 m. Each of the maps also had separate starting and destination points of the paths, as presented in Table 1; the planner also required the determination of the initial and final poses, and the pi/2 pose was selected.

Thus, thanks to the planners, the robot's movement has been programmed in such a way that it is possible to avoid physical obstacles in the field. These are illustrated on the maps in section Results.

The Simulink's simulation model was used to program the robot's motion. The diagram is shown in Figure 7.

In the experiments, the HIL simulation method was used. The paths were given using a waypoints matrix. The robot was controlled by a Pure Pursuit path-tracking algorithm. During the execution of the specified paths, in real time, the data were recorded from the encoders and the gyroscope.

The figure shows the Simulink diagram, which allowed the QBot 2e mobile robot to drive over the waypoints designated by the traffic planning algorithms. The ride started 5 seconds after the hardware simulation was started, because this is the time required for the correct calibration of the digital gyroscope placed on the QBot 2e robot; then, the robot was moving in the HIL loop. The travel time depended on the route length.

Section 1 is responsible for the 5-second delay in starting the HIL hardware simulation. The path block sends the designated waypoints from planners to the block responsible for moving the robot to the next waypoint using the Pure Pursuit algorithm.

Section 2 is responsible for calculating the kinematics of the QBot 2e mobile robot. The QBot 2e Inverse Kinematics block marked in Figure 7 as number 1 calculates and transmits values to the left and right wheel drive motors. The QBot 2e Forward Kinematics block marked in Figure 7 as number 2 requires left and right wheel velocities to be input; then, it calculates the linear velocities in the global coordinate frame. The QBot 2e Forward Kinematics (data from Gyro) block marked in Figure 7 as number 3, as with the previous

one, uses the robot's kinematic equations to calculate the pose of QBot 2e, which is required for the Pure Pursuit algorithm to continue operation. In addition, it requires information from the robot's gyroscope, i.e., the actual Theta angle (in radians).

Section 3 is responsible for recording the kinematics and gyro indications for later analysis.

Under the Simulink diagram 7, there are also blocks responsible for collecting data on the energy used; they are presented in Figure 8.

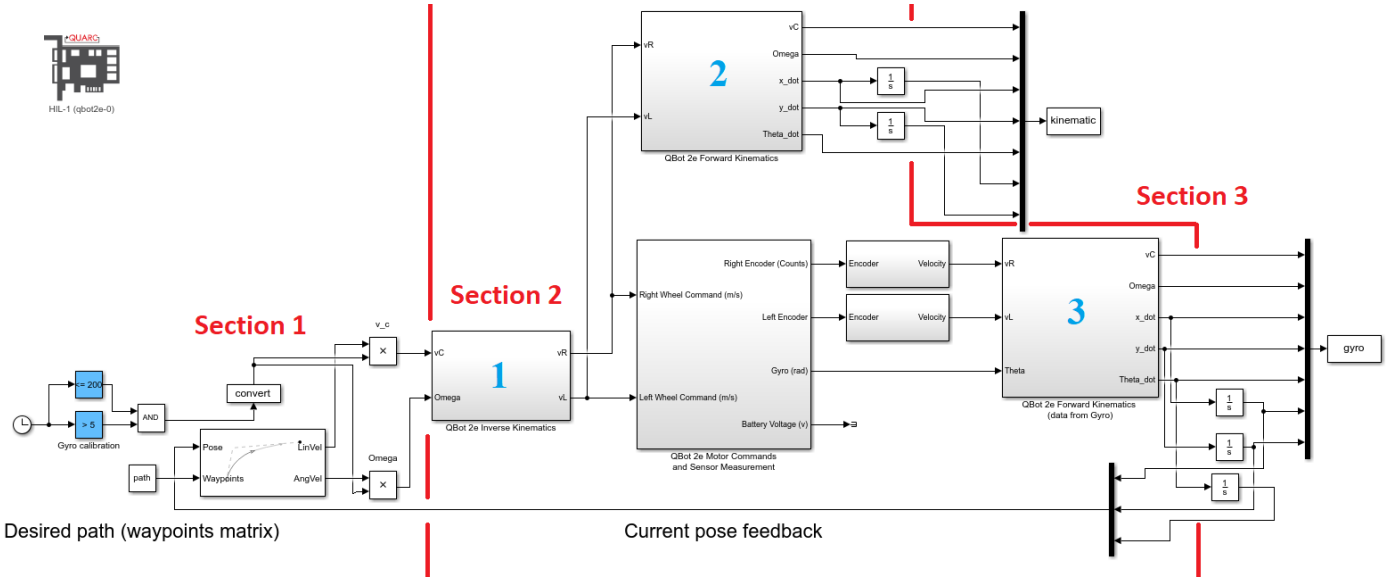

Figure 7. Simulink diagram used in a hardware in the loop simulation to investigate of QBot 2e localization algorithms.

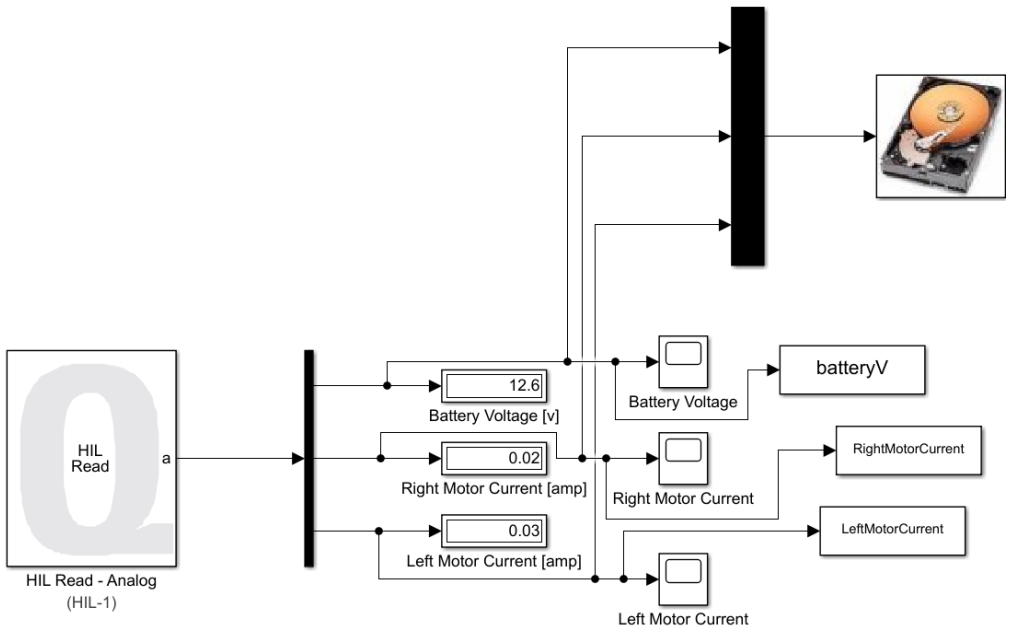

Figure 8. Simulink diagram for recording of the QBot 2e battery usage.

While driving through each of the three maps with three different generated traffic paths, during the trip, data on the electricity consumed by the motors were collected, and the distances traveled on the route were measured from the right and left encoders. The state record was registered to the workspace as well as to a file (a block with a hard drive).

The HIL Read-Analog block reads specific analog channels, and data are read in real time. The To Host File block (with hard drive) can be used to save time and input data in various data file formats or to save input data as a video file. The block saves scalar or vector input data to the matrix in a file, and it accepts signals of any of the built-in data types in Simulink. Other blocks are responsible for forwarding the signals, saving to the workspace and showing the read data during operation.

The final stage of the study was the off-line analysis of the HIL simulation results. In the analysis, the following parameters were related to energy. Recall that in the work energy, consumption was measured only by motors. Energy consumption by sensors and electronics was not taken into account, recognizing that this amount is negligible and similar for each of the routes covered.

The recorded data on the voltage level of the U battery and the electric current were separate for the left wheel motor I_l and the right wheel motor I_r. These data were used to calculate electrical power P (work) per unit time, which is defined for us as a second.

$$P = U * (I_l + I_r), \tag{9}$$

Thanks to Formula (9), the results of the power used during the entire drive were obtained; for each of the paths, they are presented in section Results.

$$E = \sum_{i=1}^{100} \frac{P_i}{100}, \tag{10}$$

where P_i—power consummed by motors for one sample period.

The measurements were carried out by sampling the values of the current supplying the right and left motors. At the same time, the voltage of the battery supplying the motors was sampled. The sample frequency equals 100 Hz. Energy consumption by sensors and electronics was not taken into account, recognizing that this amount is negligible and similar for each of the routes covered. The aim of the work was to determine the drive energy consumtion, because it strictly depends on the length of the paths and the complexity of maneuvers between obstacles. From Equation (10), E the average power was obtained in one second. The results for each of the drives are presented in section Results. The results of the analysis are presented in the further part of the paper.

$$\sigma = \sqrt{\frac{\sum (\overline{E} - P_i)^2}{100}}, \tag{11}$$

where σ is a standard deviation.

The values of standard deviations for each of the tested obstacle maps are presented in the further part of the paper.

4. Results

The selected results of tests carried out in accordance with the methodology described above are shown in Figures 9–44 and Table 2. Figures 9, 12, 15, 18, 21, 24, 27, 30, and 33 show the obstacle maps. Blue indicates the operation of path-planning algorithms (busy grid analysis), red indicates the found path from the given starting point to the destination. The travel paths in the HIL loop are depicted with the yellow path. Figures 10, 13, 16, 19, 22, 25, 28, 31, and 34 show the linear speed for the paths traveled by the robot determined on the earlier maps together with the kinematics model. Figures 11, 14, 17, 20, 23, 26, 29, 32, and 35 show the angular speed for the paths covered by the robot, determined on the previous maps, together with the kinematics model.

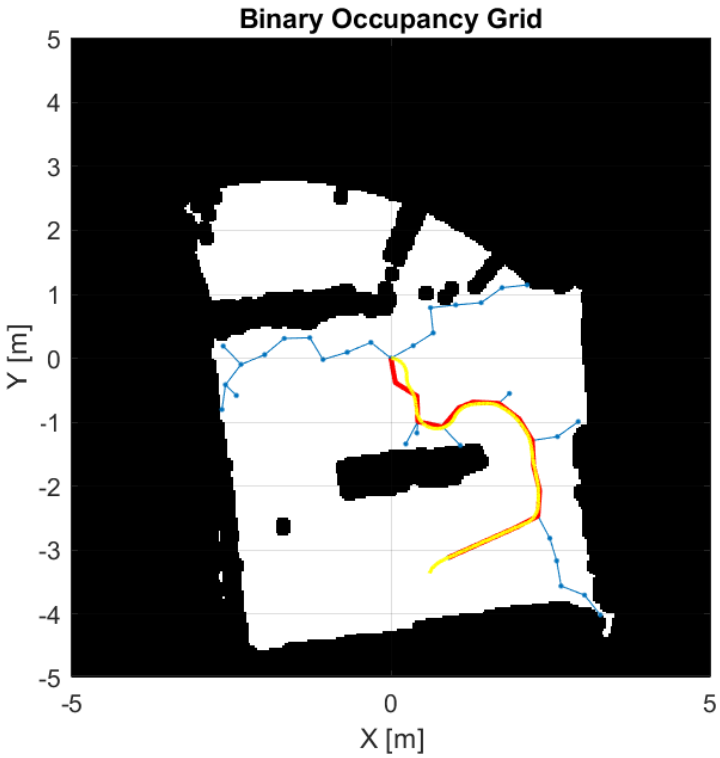

Figure 9. Desired waypoints obtained with RRT algorithm and path obtained from odometry data.

4.1. Results for First Map

The first generated map with the mapping of the path using the RRT algorithm is presented below. In Figures 10 and 11, the linear and angular velocity of the robot were presented. They were determined on the basis of odometric data and the kinematic model.

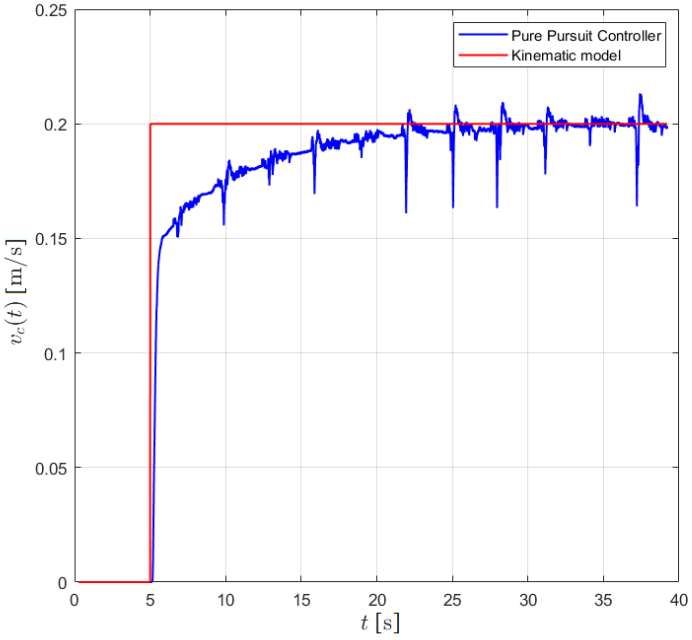

Figure 10. Linear velocity $v_c\ (t)$ of the robot.

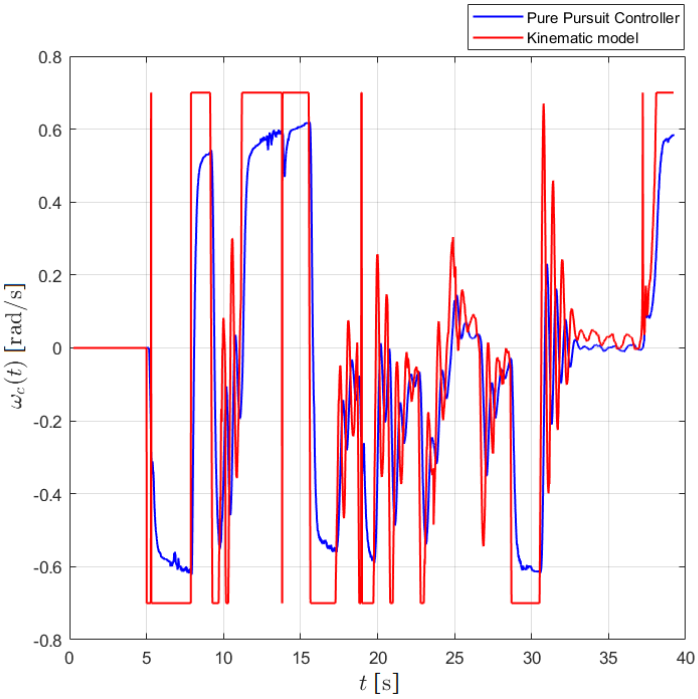

Figure 11. Angular velocity $\omega_c\ (t)$ of the QBot 2e robot.

The first generated map with the mapping of the path using the RRT star algorithm is presented below. In Figures 13 and 14, the linear and angular velocity of the robot are presented. They were determined on the basis of odometric data and the kinematic model.

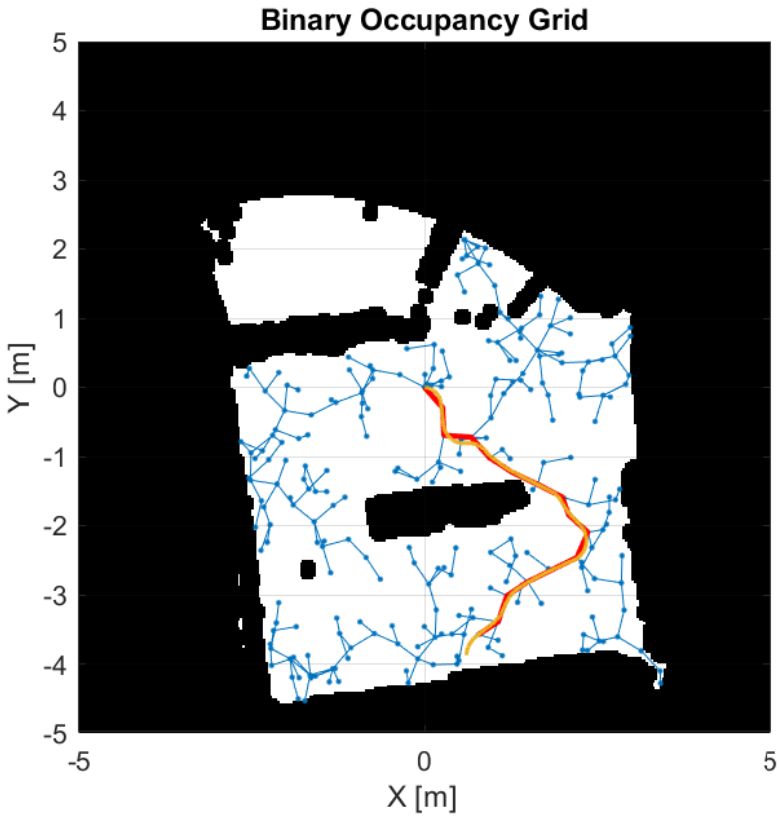

Figure 12. Desired waypoints obtained with RRT star algorithm and path obtained from odometry data.

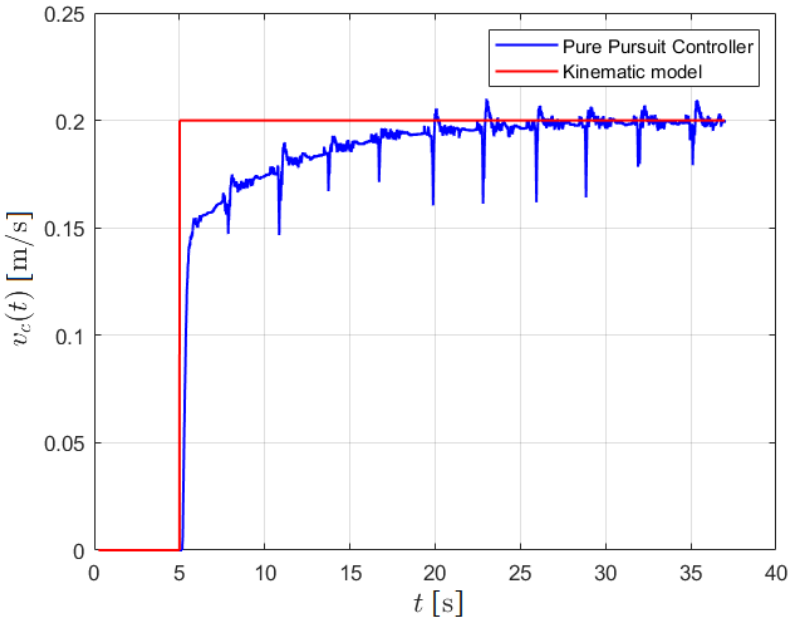

Figure 13. Linear velocity $v_c\ (t)$ of the robot.

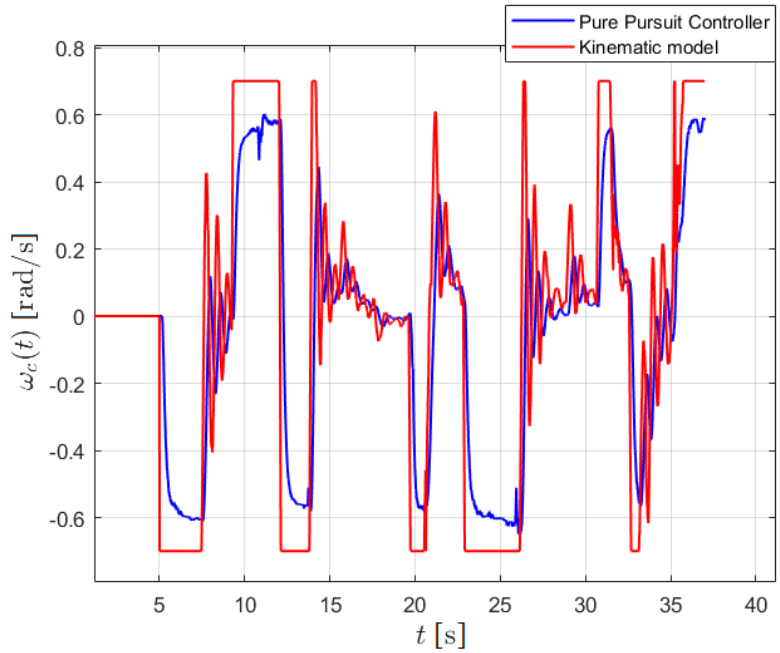

Figure 14. Angular velocity $\omega_c\ (t)$ of the QBot 2e robot.

The first generated map with the mapping of the path using the HybridA star algorithm is presented below. Figures 16 and 17 present the linear and angular velocity of the robot. They were determined on the basis of odometric data and the kinematic model.

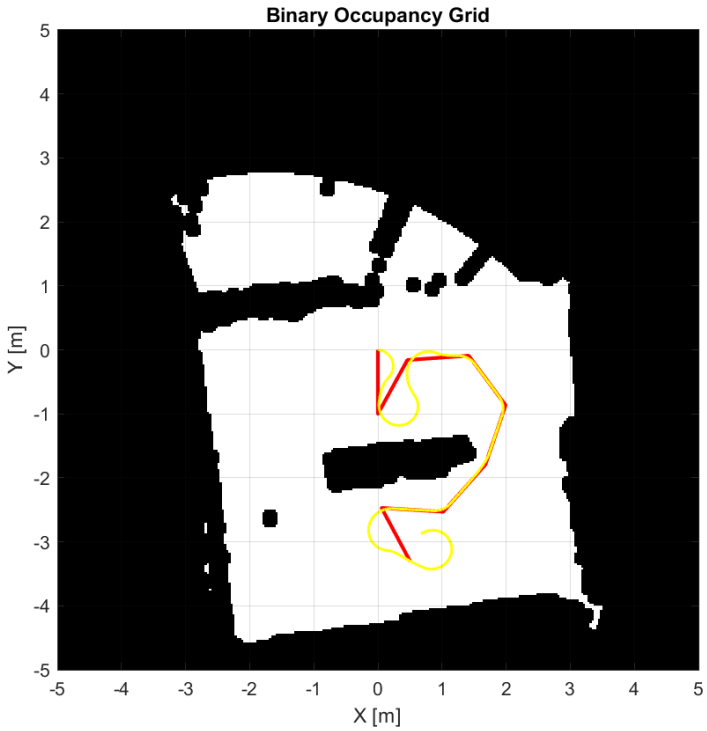

Figure 15. Desired waypoints obtained with HybridA star algorithm and path obtained from odometry data.

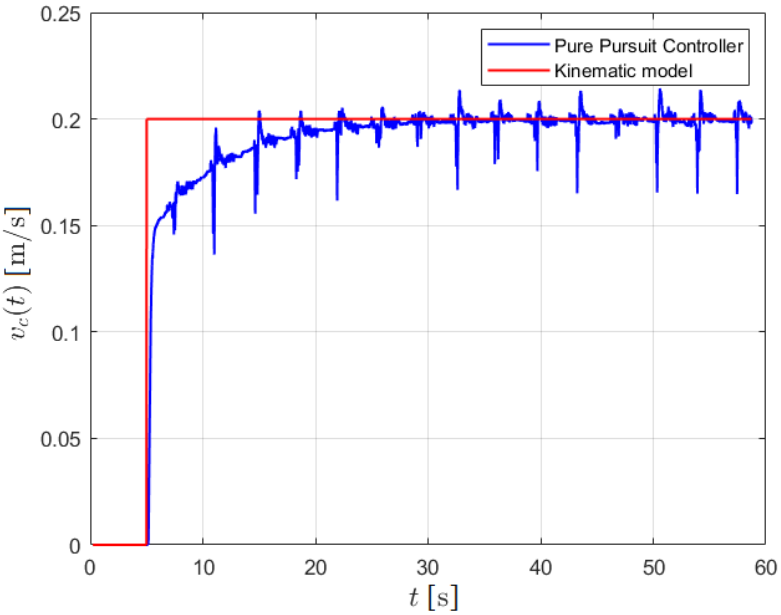

Figure 16. Linear velocity $v_c\ (t)$ of the robot.

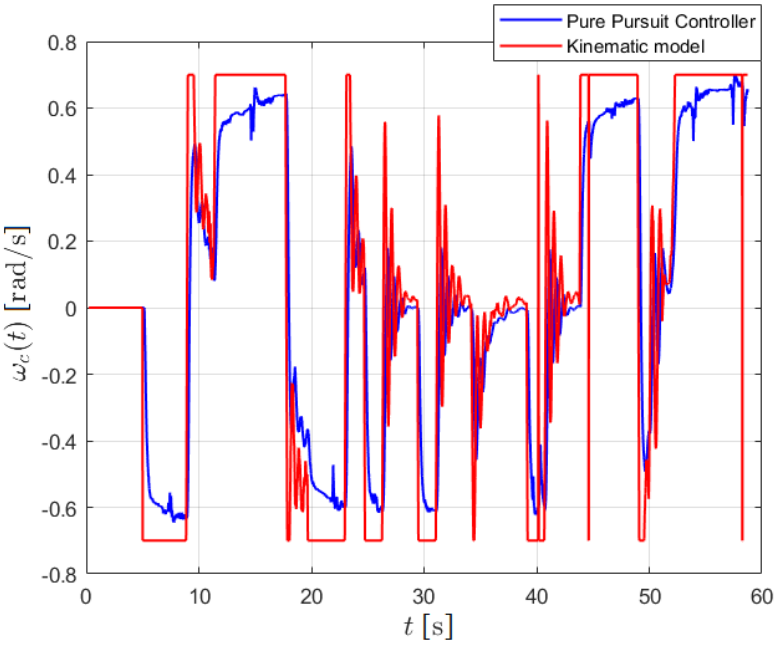

Figure 17. Angular velocity $\omega_c\ (t)$ of the QBot 2e robot.

4.2. Results for Second Map

The second generated map with the mapping of the path using the RRT algorithm is presented below. In Figures 19 and 20, the linear and angular velocity of the robot were presented. They were determined on the basis of odometric data and the kinematic model.

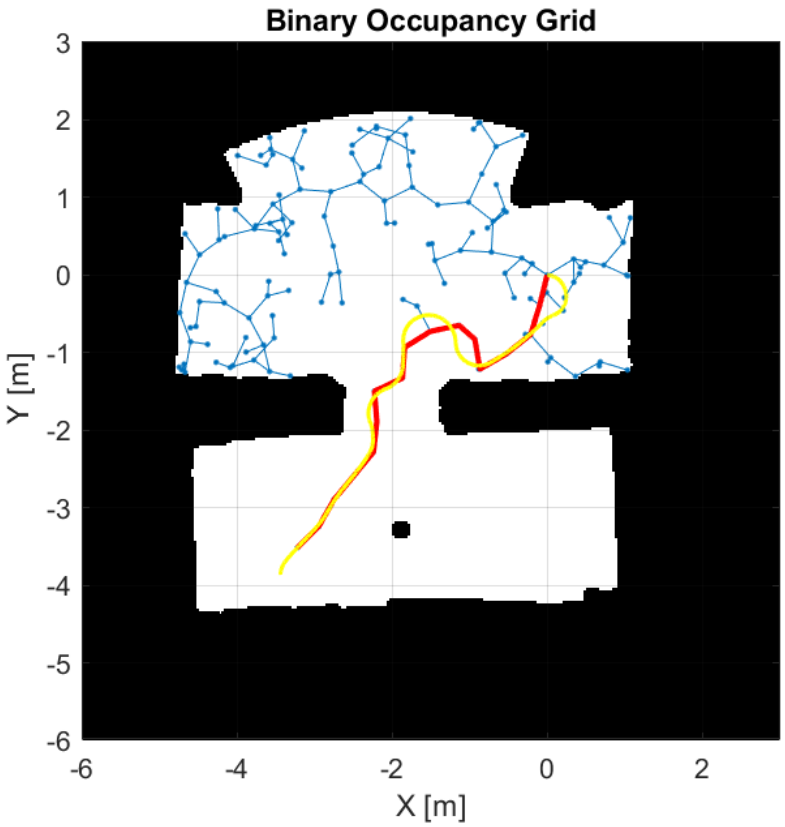

Figure 18. Desired waypoints obtained with RRT algorithm and path obtained from odometry data.

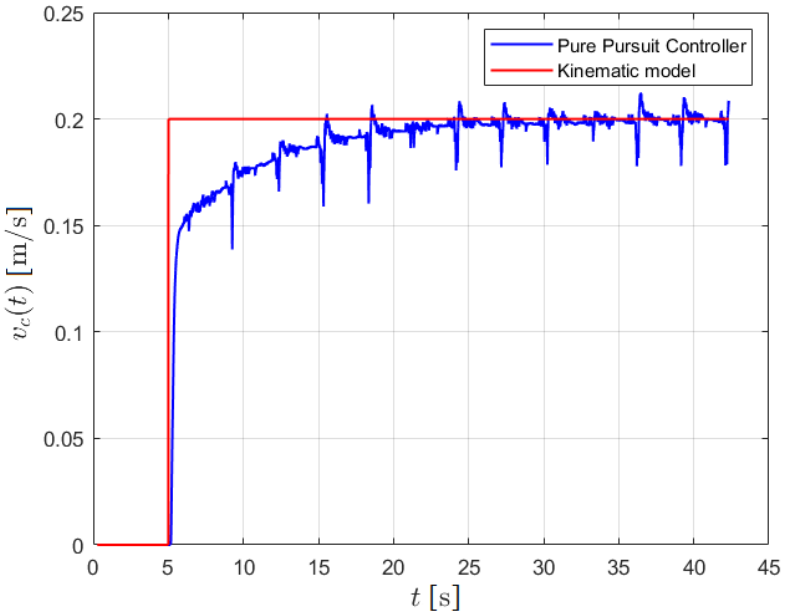

Figure 19. Linear velocity $v_c\ (t)$ of the robot.

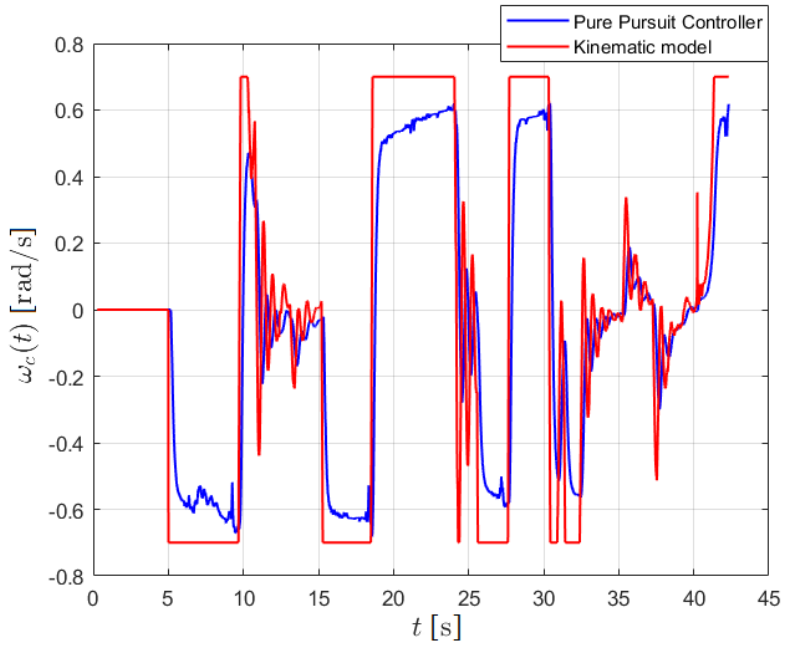

Figure 20. Angular velocity $\omega_c\ (t)$ of the QBot 2e robot.

The second generated map with the mapping of the path using the RRT star algorithm is presented below. In Figures 22 and 23 the linear and angular velocity of the robot were presented. They were determined on the basis of odometric data and the kinematic model.

The second generated map with the mapping of the path using the HybridA star algorithm is presented below. Figures 25 and 26 present the linear and angular velocity of the robot. They were determined on the basis of odometric data and the kinematic model.

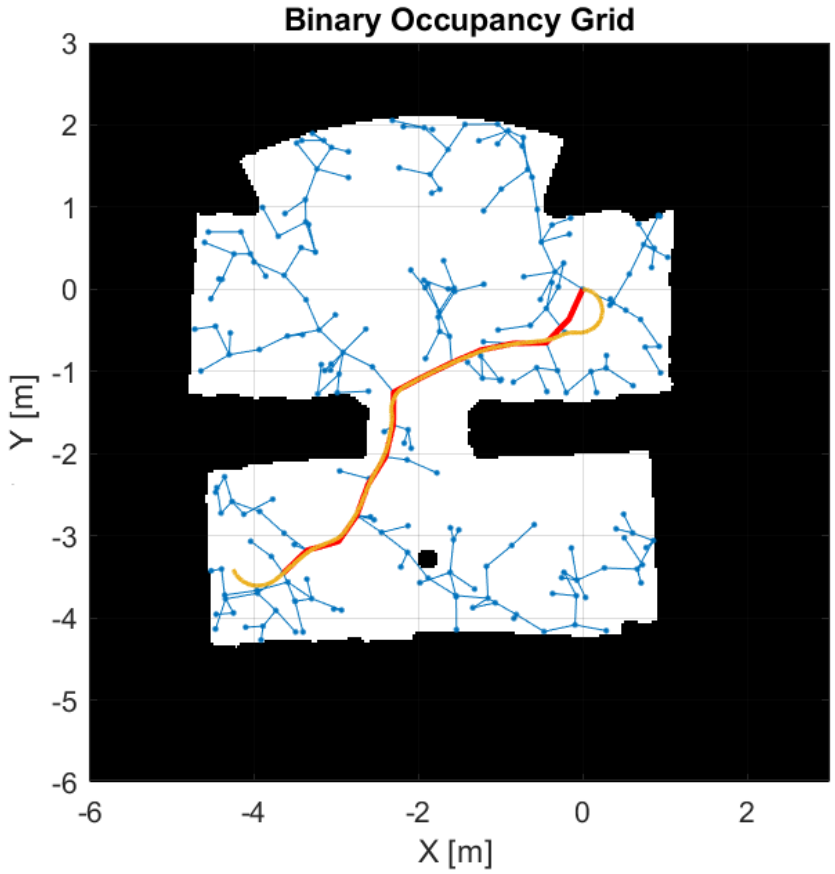

Figure 21. Desired waypoints obtained with RRT star algorithm and path obtained from odometry data.

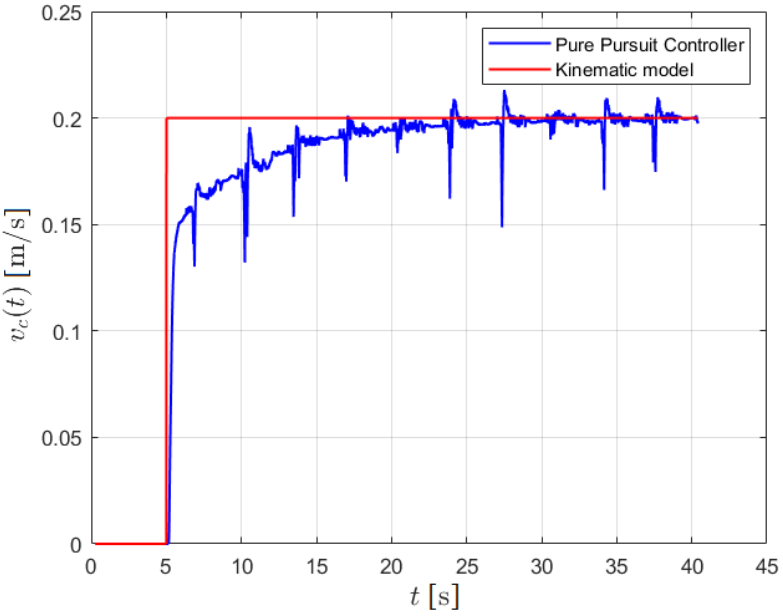

Figure 22. Linear velocity $v_c\ (t)$ of the robot.

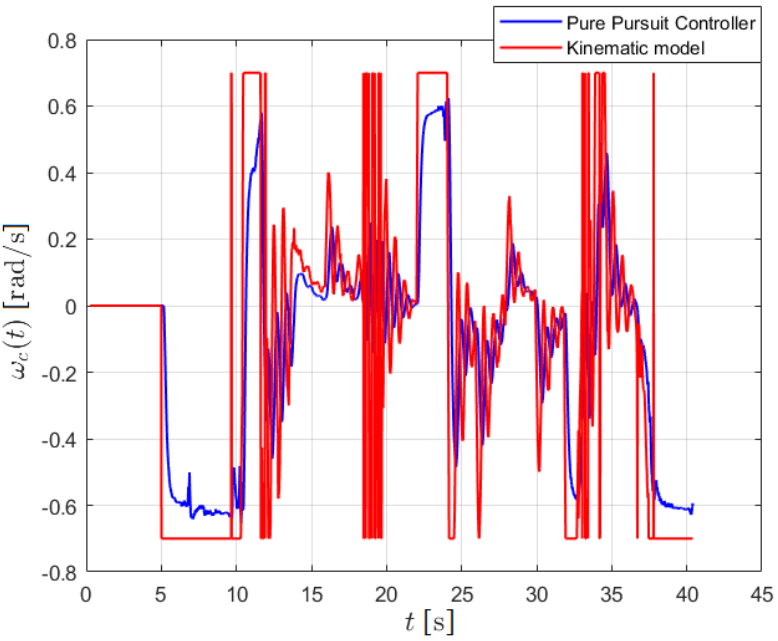

Figure 23. Angular rate $\omega_c\ (t)$ of the QBot 2e robot.

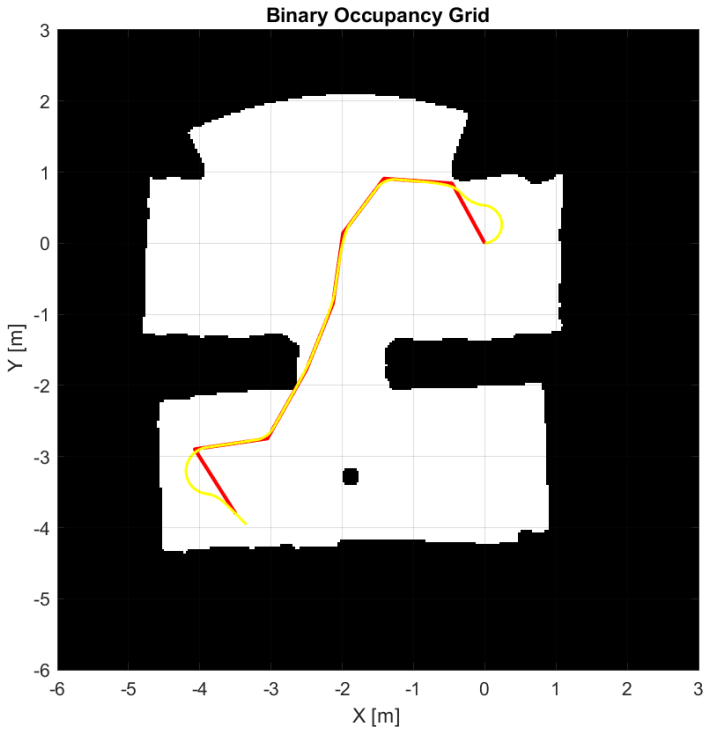

Figure 24. Desired waypoints obtained with HybridA star algorithm and path obtained from odometry data.

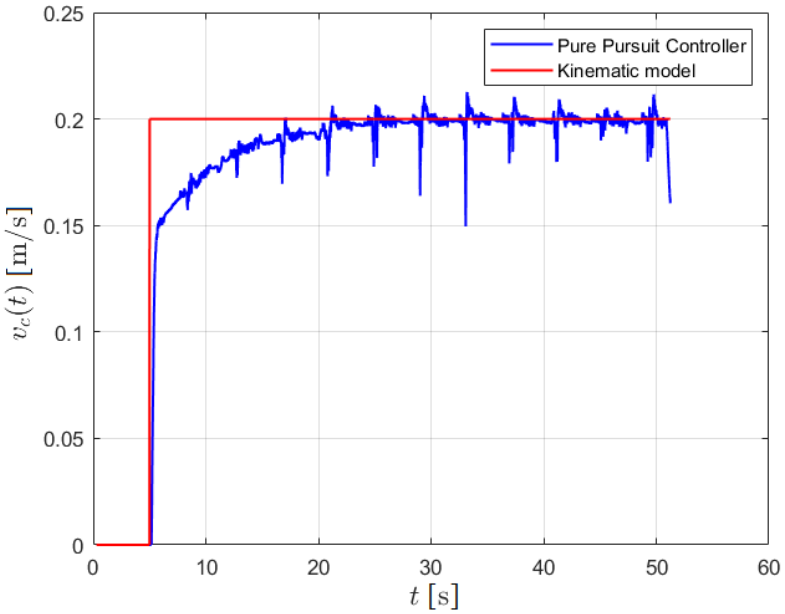

Figure 25. Linear velocity $v_c\ (t)$ of the robot.

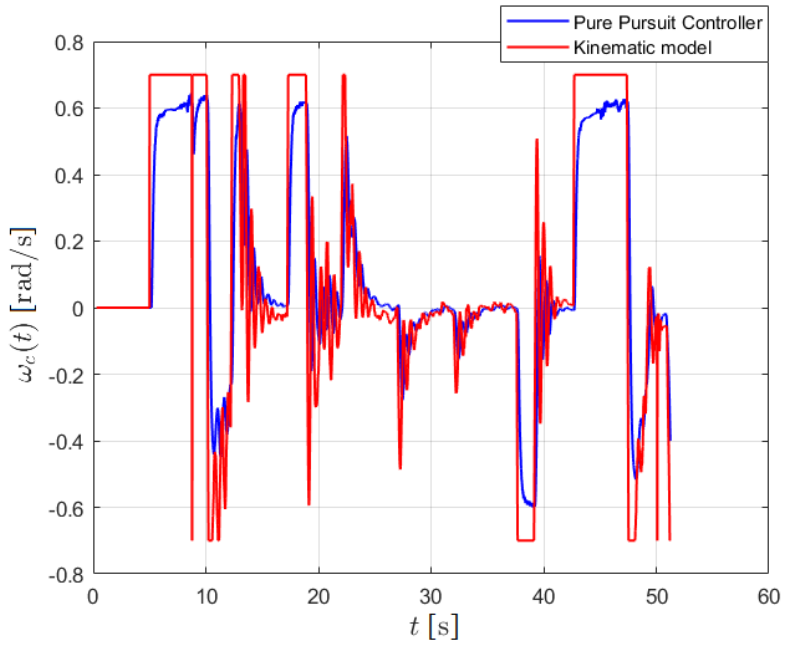

Figure 26. Angular velocity $\omega_c(t)$ of the QBot 2e robot.

4.3. Results for Third Map

The third generated map with the mapping of the path using the RRT algorithm is presented below. Figures 28 and 29 present the linear and angular velocity of the robot. They were determined on the basis of odometric data and the kinematic model.

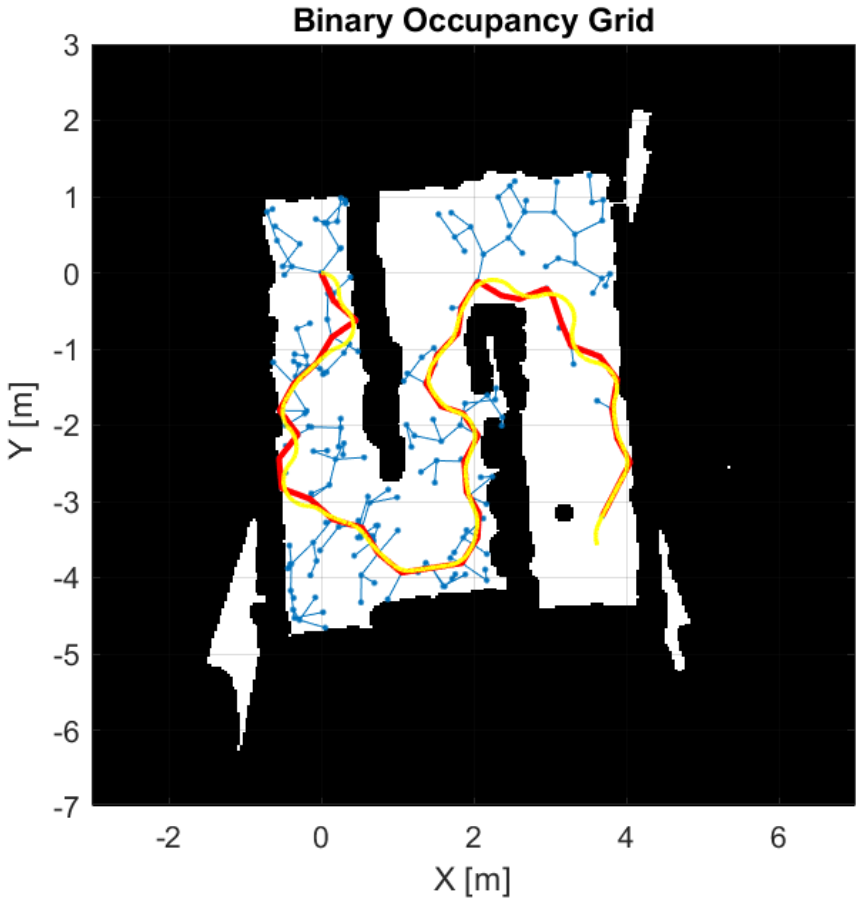

Figure 27. Desired waypoints obtained with RRT algorithm and path obtained from odometry data.

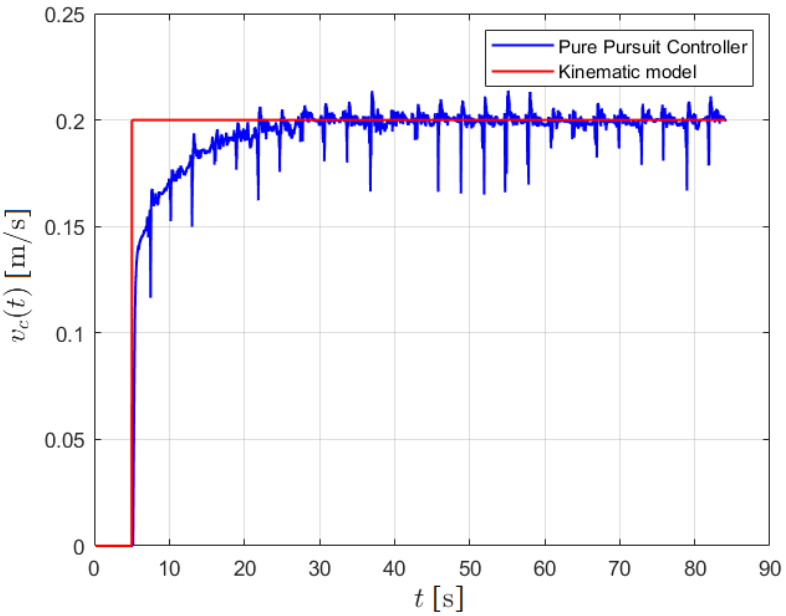

Figure 28. Linear velocity $v_c\ (t)$ of the robot.

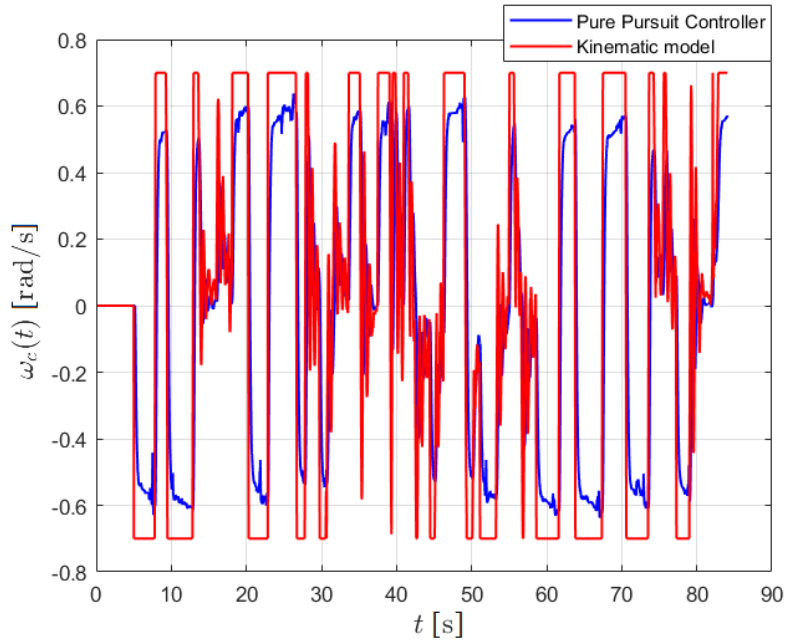

Figure 29. Angular velocity $\omega_c\ (t)$ of the QBot 2e robot.

The third generated map with the mapping of the path using the RRT star algorithm is presented below. Figures 31 and 32 present the linear and angular velocity of the robot. They were determined on the basis of odometric data and the kinematic model.

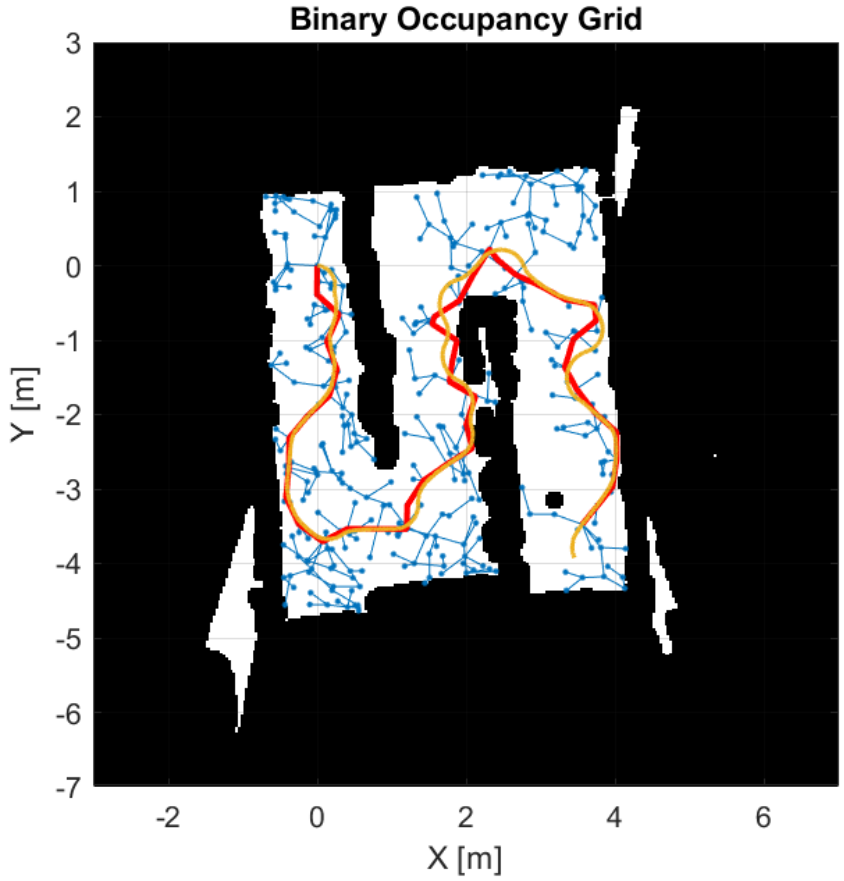

Figure 30. Desired waypoints obtained with RRT star algorithm and path obtained from odometry data.

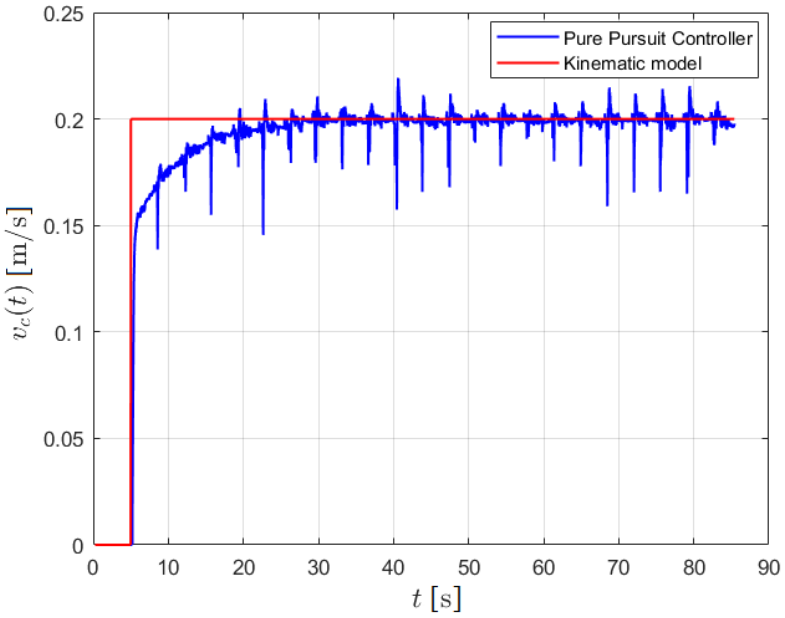

Figure 31. Linear velocity $v_c\ (t)$ of the robot.

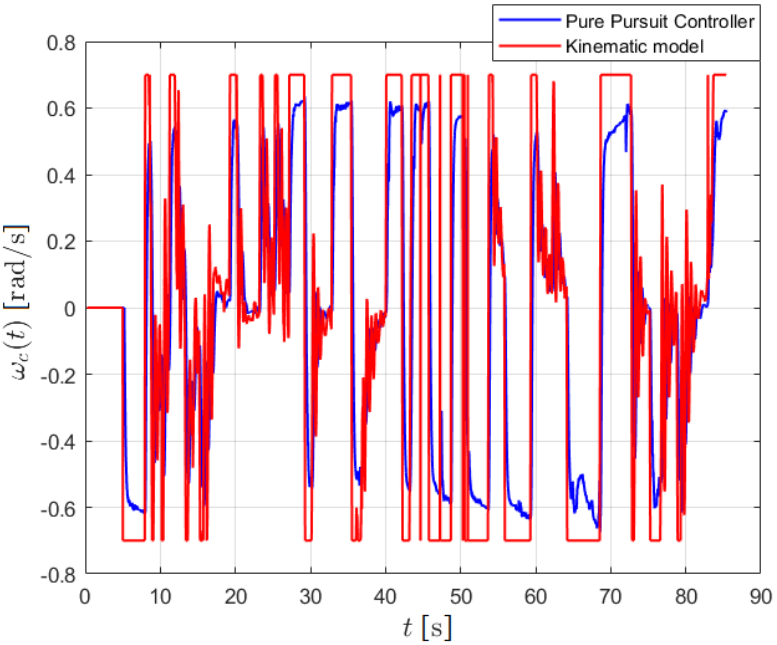

Figure 32. Angular velocity $\omega_c\ (t)$ of the QBot 2e robot.

The third generated map with the mapping of the path using the RRT algorithm is presented below. Figures 34 and 35 present the linear and angular velocity of the robot. They were determined on the basis of odometric data and the kinematic model.

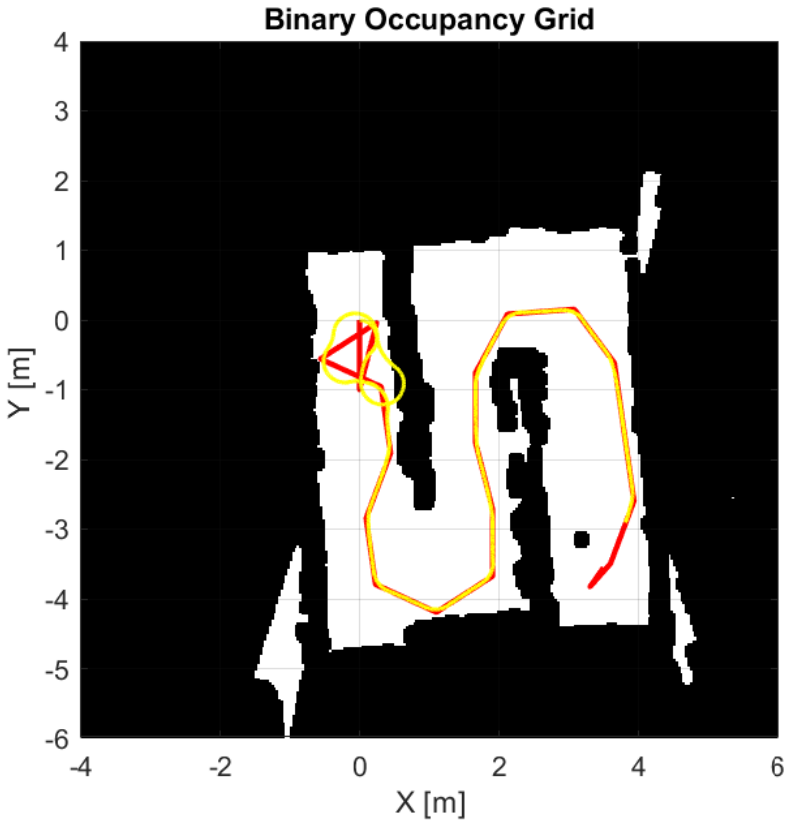

Figure 33. Desired waypoints obtained with HybridA star algorithm and path obtained from odometry data.

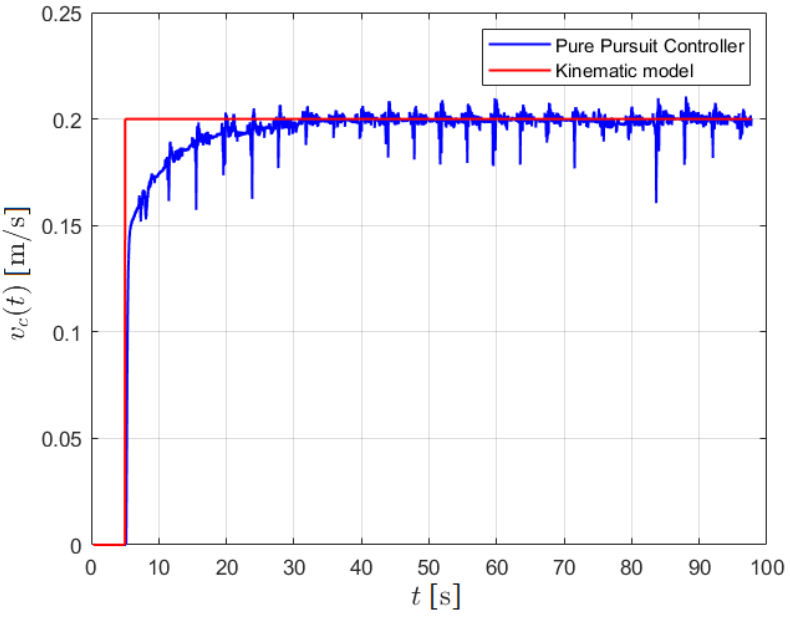

Figure 34. Linear velocity $v_c\ (t)$ of the robot.

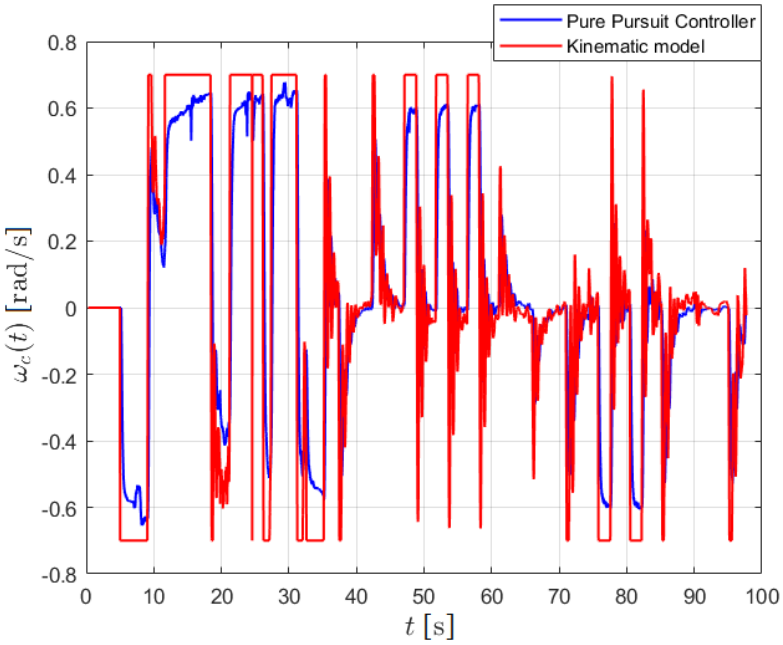

Figure 35. Angular velocity $\omega_c\ (t)$ of the QBot 2e robot.

4.4. Results for Energy

The results of the power used during the entire drive were obtained; for each of the paths, they are presented in Figures 36–44.

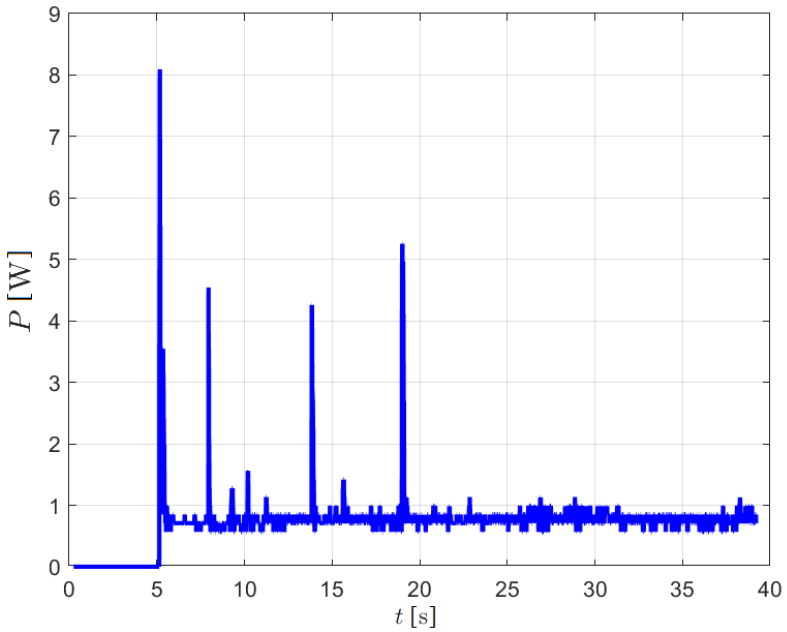

Figure 36. Power P during *Map*1 run for the RRT algorithm of the robot.

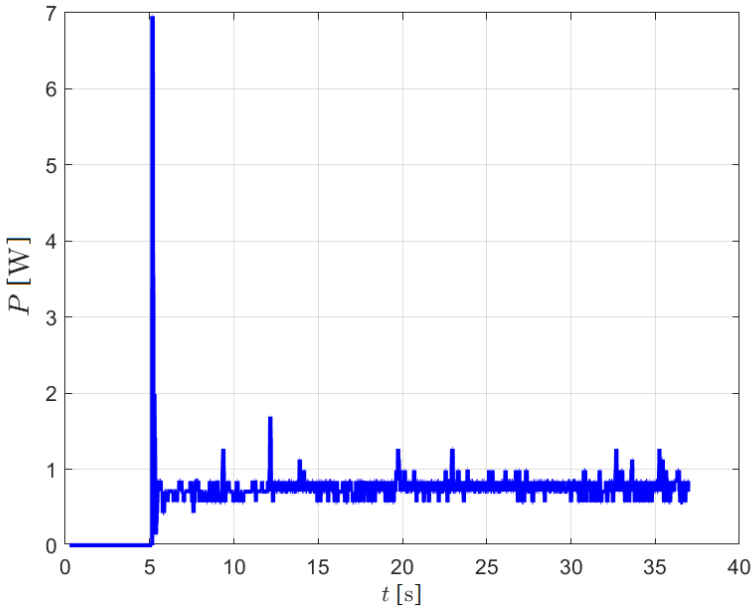

Figure 37. Power P during $Map1$ run for the RRT star algorithm of the robot.

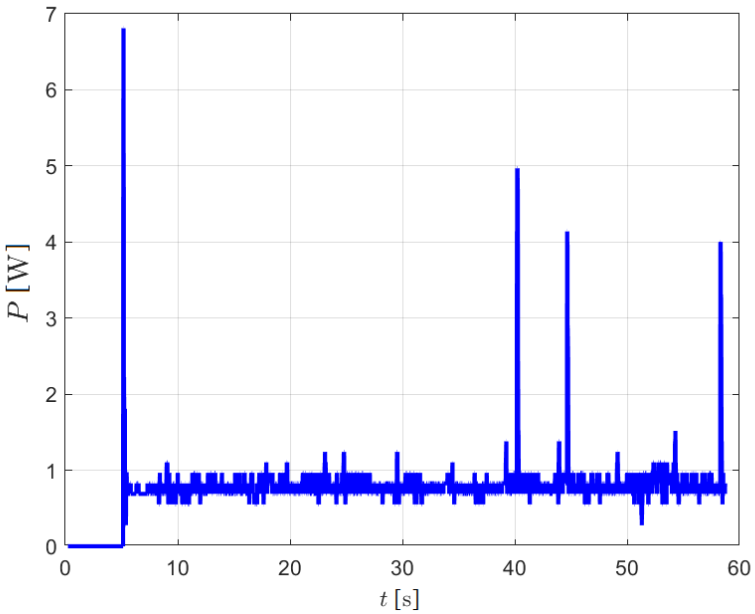

Figure 38. Power P during $Map1$ run for the HybridA star algorithm of the robot.

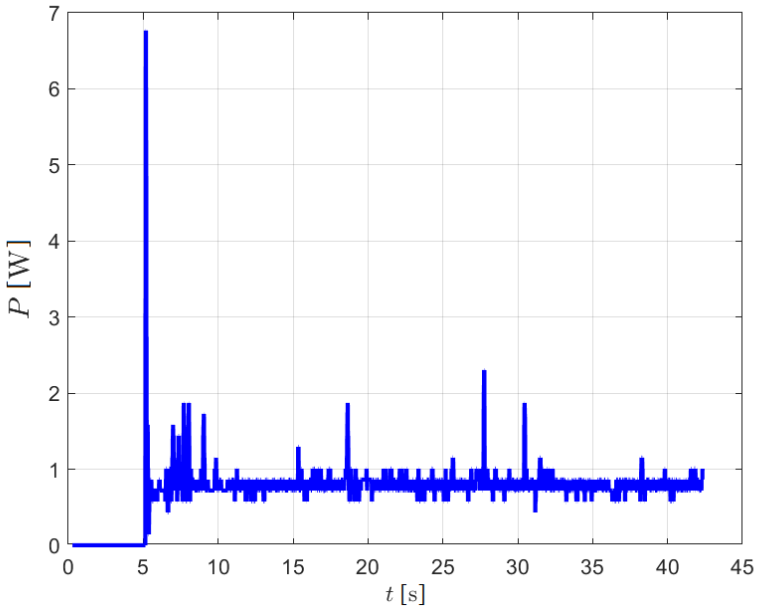

Figure 39. Power P during *Map*2 run for the RRT algorithm of the robot.

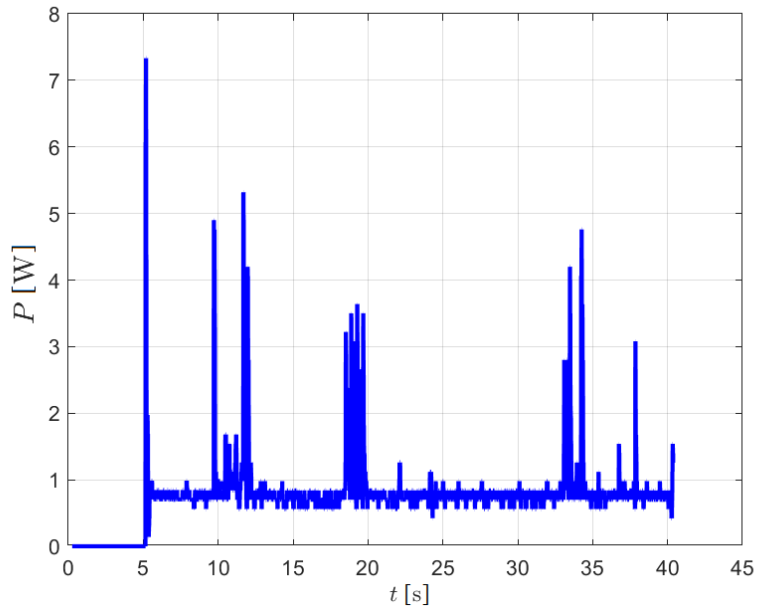

Figure 40. Power P during *Map*2 run for the RRT star algorithm of the robot.

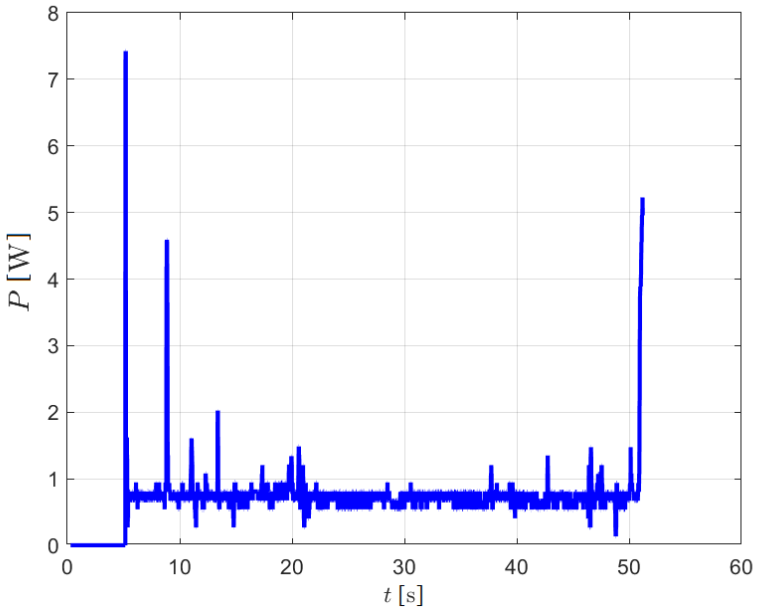

Figure 41. Power P during *Map*2 run for the HybridA star algorithm of the robot.

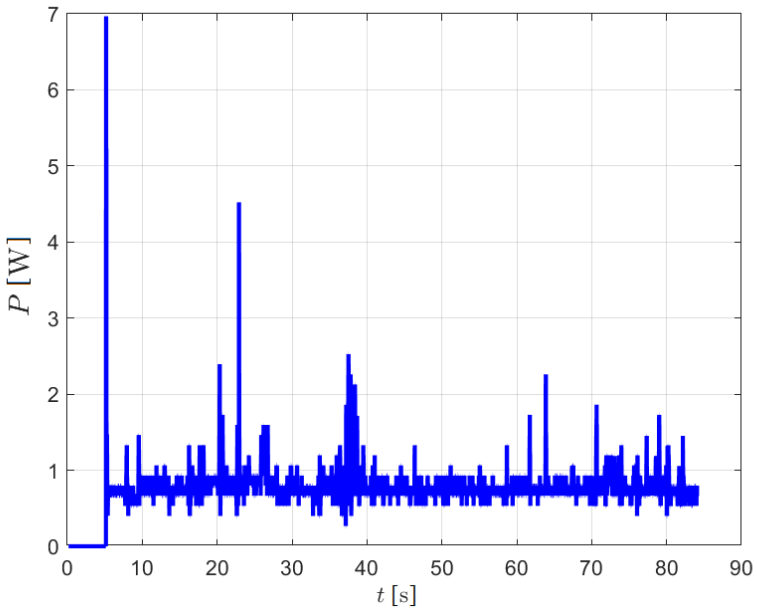

Figure 42. Power P during *Map*3 run for the RRT algorithm of the robot.

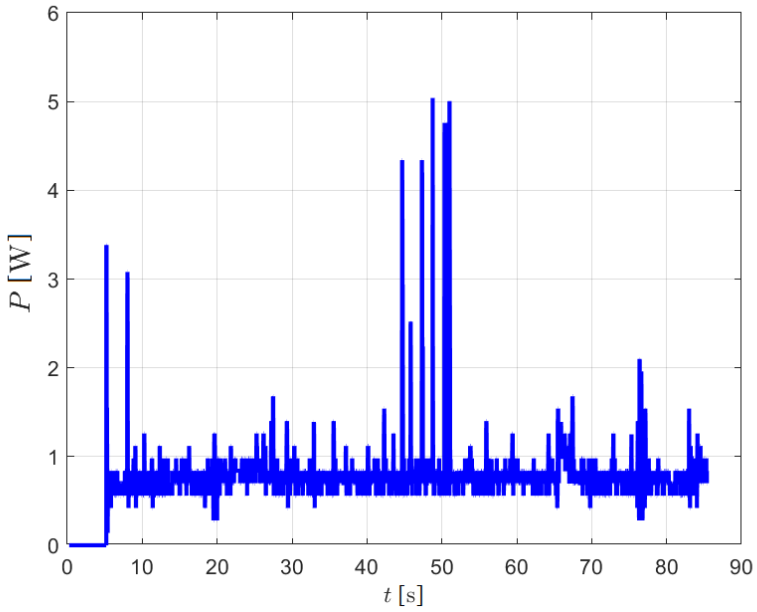

Figure 43. Power P during *Map*3 run for the RRT star algorithm of the robot.

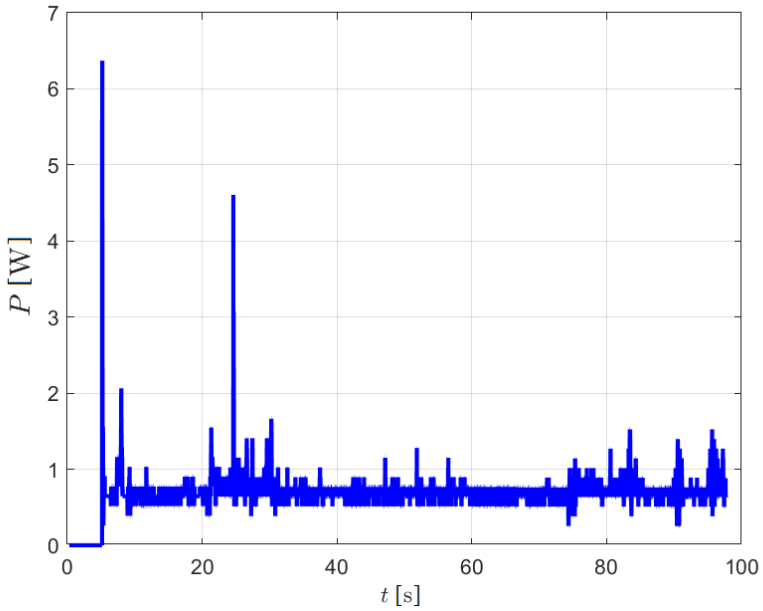

Figure 44. Power P during *Map*3 run for the HybridA star algorithm of the robot.

Table 2 below shows the results of the electricity consumed for the execution of individual trajectories. The method for calculating the value is shown in the Methodology section.

Table 2. The table shows the cost of energy used to travel the generated path.

Map	Map 1 (Ws)	Map 2 (Ws)	Map 3 (Ws)
RRT	27.61	30.26	62.35
RRT star	24.24	30.08	63.62
HybrydA star	43.45	35.36	65.90

The values of standard deviations for each of the tested obstacle maps are presented in Table 3. They were calculated using the Formula (11).

Table 3. The table shows the standard deviation of energy consumption for each obstacle map.

Standard Deviation	Map 1	Map 2	Map 3
σ	8375	2448	1469

5. Discussion

Three map layouts were tested in the paper: the first map had one obstacle in the middle of the working space, the second map reflected the division into two rooms, and the third and last map was a maze with narrow passages.

For the arrangement of obstacles in the first map, it can be seen that the algorithms had the option of choosing two paths to the goal on the left or right side of the obstacle. It has been observed in Figure 12 showing the RRT star algorithm that it is the best traced path, because by applying the cost, it directly goes to the designated destination and is the shortest route. Additionally, we can see in Table 2 that the path used the least amount of energy to reach the destination. The HybridA star algorithm performed the worst in this setup, as it moves closer and farther away from the mission target, as shown in Figure 15.

For the second set of obstacles, consisting of two rooms, it can be seen in Figures 18, 21 and 24 that the algorithm has only one of the possibilities of taking a course on the target and selects it first. Again, the best algorithm turns out to be RRT star (Figure 21), which directs itself to the target coordinates without unnecessary map keying. The RRT algorithm also showed a good way to the goal, but in Figure 18, we can see problems with going through the bottleneck between the rooms. After analyzing the results in Table 2, it can be seen that both algorithms have similar energy consumption. The HybridA star algorithm performed the worst; in Figure 24, we can see that it chooses the opposite direction than the one to the goal. The power consumption is different from other algorithms.

For the third set of obstacles, depicting a maze with three narrow corridors, here, all three algorithms RRT for Map (Figure 27), RRT star for Map (Figure 30) and HybridA star for Map (Figure 33) developed identical paths to reach the destination. Only the HybridA star algorithm had problems at the end of the path, as seen in Figure 33, trying to reach the target coordinates too accurately (it requires changing its settings in the future). Thus, when passing the generated path through HybridA star, the QBot 2e robot invaded and moved the placed obstacles; see Figure 33. The travel path is marked yellow on the map. The creation of the map and subsequent generation of occupancy paths requires a safety margin. Ideally, it should be a minimum of half the diameter of the mobile robot, taking the notation in Figure 4; this would be the equation $d/2$. In the tested system of obstacles, the lowest energy consumption (Table 2) had the RRT star algorithm, while the HybridA star used the most energy to reach its destination.

Analyzing the results of the tests for all three paths and hardware simulations, it can be concluded that the determination of the position of the robot using kinematic Equation (8) is subject to a large error. Odometer-based location is much more accurate. The mathematical model of kinematics does not take into account the dynamic properties of the mobile robot, i.e. inertia, and errors in the calculations of the kinematics result from it. An additional problem is the slippage of the robot's wheels, which causes the encoder to register more wheel rotations, which is not related to the actual change of the position of the robot. We emphasize that the research was carried out in a laboratory in the Autonomous Vehicles Research Studio workspace from QUANSER. Thus, during the tests, wheel slippage was limited with the use of anti-slip mats; additionally, weather conditions did not hinder the execution of the designated paths to reach the destination.

Changes in the Pure Pursuit path-tracking algorithm parameters were not taken into account during the study. The same values were set for all passes. The Pure Pursuit motion controller turned out to be overestimated; certain values such as Look a Head were set too far away from the robot. This resulted in dynamic changes in encoder values, which contribute to the increase in energy consumption, as shown in

Figures 11, 14, 17, 20, 23, 26, 29, 32, and 35. In addition, comparing them with Figures 36–44, a significant increase in energy consumption can be deduced for rapid changes in direction of travel. The Pure Pursuit algorithm should be adapted to the QBot 2e mobile robot, which would reduce the dynamics of the wheel movement and further reduce energy consumption. It was precisely the mismatch of the Look a Head parameter in Pure Pursuit that caused significant jumps in the encoders, which contribute to the increase in energy consumption. Complementing in Figures 36–44, the initial spikes in energy consumption are measured at the start of the movement; this is due to the desire to meet the velocities generated by the kinematics model, as observed in Figures 10, 13, 16, 19, 22, 25, 28, 31 and 34. The kinematics model does not take into account the dynamic properties of the QBot 2e mobile robot, such as, for example, the weight of the vehicle or air resistance.

The calculated standard deviation of the population from the Formula (11) in Table 3 shows that the seemingly simplest maps (containing one obstacle, such as Figure 9) are in fact the most difficult for the path generation algorithms, because the map has a large number of possibilities to reach the destination and thus most calculate the most to choose the ways to reach the finish line. The algorithm must decide whether the obstacle to be avoided is to be on the right or left side. While seemingly difficult maps, such as Figure 27, are easy to analyze by path-planning algorithms, in the case of a large number of obstacles and a more complicated track, it turns out that the path selection options are significantly limited. With the map of the labyrinth, we can see that the paths drawn by the algorithms are the same, map in Figures 27, 30 and 33, because in this case, there is only one way to arrive at the destination without avoiding obstacles. Thus, the results from Table 3 are confirmed, where the highest standard deviation of energy consumption is visible on map 1. It is much smaller on map 2, where the way to reach the destination was limited by using a narrow passage. The smallest deviation was recorded by the route from map 3: in this case, only one route is possible to reach the destination.

Summarizing, increasing the accuracy of the generated path with greater consideration of the robot's homologous and dynamic constraints reduces its energy consumption. The use of weights in the occupancy grid in the RRT star algorithm makes it possible to find the shortest route to the destination, and thus the energy cost is the lowest. This is noticeable in the HybridA star algorithm, where the path significantly deviates from the optimal one.

6. Conclusions

The conducted experiments have proved that an important element of path planning is the selection of the algorithm that generates it and the skilful selection of the path-planning model, which can significantly reduce the energy consumption costs. However, its significance decreases in relation to the alleged complexity of the obstacle map.

It is important to analyze more path-planning algorithms on apparently simpler maps with few obstacles and walls.

Changes in the parameters of the Pure Pursuit regulator on energy consumption were not examined in the work. The effect of selecting the optimal path-planning algorithm is insignificant energy savings, which, with battery limitations, translate slightly into the working time and field range of operation and the possibility of reaching the destination without the need to recharge.

With the tested maps and the QBot 2e robot, the best algorithm turned out to be RRT star, which generated the best path in terms of energy consumption with the obstacle maps 1 and 2. In the case of map 3, the choice of path-planning algorithm is not important.

Finally, it should be emphasized that the conclusions drawn are limited to the number of algorithms, generated obstacle maps and various types of mobile robots. The tested algorithms may differ significantly with other maps and the robots used for the experiments.

Author Contributions: Conceptualization, A.R. and S.D.; methodology, A.R. and S.D.; software, A.R.; validation, A.R.; formal analysis, S.D.; investigation, A.R.; resources, A.R.; data curation, A.R.; writing—original draft preparation, A.R.; writing—review and editing, A.R. and S.D.; visualization,

A.R.; supervision, S.D.; project administration, A.R.; funding acquisition, A.R. and S.D.; All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Clue, A.; Le, R.; Wang, J.; Ahn, I.S. Distributed Vision-Based Target Tracking Control Using Multiple Mobile Robots. In Proceedings of the IEEE International Conference on Electro/Information Technology (EIT), Rochester, MI, USA, 3–5 May 2018; pp. 0471–0476.
2. Trojnacki, M.; Szynkarczyk, P.; Andrzejuk, A. Tendencje rozwoju mobilnych robotów lądowych. *Pomiary Autom. Robot.* **2008**, *6*, 11–14.
3. Chernousko, F.L. Locomotion Principles for Mobile Robotic Systems. *Procedia Comput. Sci.* **2017**, *103*, 613–617. [CrossRef]
4. Machado, T.; Malheiro, T.; Monteiro, S.; Erlhagen, W.; Bicho, E. Attractor dynamics approach to joint transportation by autonomous robots: theory, implementation and validation on the factory floor. *Auton. Robots* **2019**, *43*, 589–610. [CrossRef]
5. Yamaguchi, H.; Nishijima, A.; Kawakami, A. Control of two manipulation points of a cooperative transportation system with two car-like vehicles following parametric curve paths. *Robot. Auton. Syst.* **2015**, *63*, 165–178. [CrossRef]
6. Dudzik, S.; Podsiedlik, A.; Rapalski, A. Test stand to design control algorithms for mobile robots. *Prz. Elektrotech.* **2021**, *3*, 97. [CrossRef]
7. Coulter, R. *Implementation of the Pure Pursuit Path Tracking Algorithm*; Carnegie Mellon University: Pittsburgh, PA, USA, 1990.
8. Konstantinos, S.; Ibrahim, A.; DMiroslav, D. Kinematics of four-wheel-steering vehicles. *Forsch. Ing.* **2001**, *66*, 211–216.
9. Ishigami, G.; Nagatani, K.;Yoshida, K. Path Following Control with Slip Compensation on Loose Soil for Exploration Rover. In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 9–13 October 2006; pp. 5552–5557.
10. Kevin, M.L.; Frank, C.P. *Modern Robotics: Mechanics, Planning, and Control*, 1st ed.; Cambridge University Press: Cambridge, MA, USA, 2017.
11. Gharajeha, M.S.; Jondb, H.B. Hybrid Global Positioning System-Adaptive Neuro-Fuzzy Inference System based autonomous mobile robot navigation. *Robot. Auton. Syst.*, **2020**, *134*, 103669. [CrossRef]
12. Jadidi, M.G.; Miro, J.V.; Dissanayake, G. Gaussian processes autonomous mapping and exploration for range-sensing mobile robots. *Auton. Robot.* **2018**, *42*, 273–290.
13. Niewola, A.; Leszek Podsędkowski, L. PSD—Probabilistic algorithm for mobile robot 6D localization without natural and artificial landmarks based on 2.5D map and a new type of laser scanner in GPS-denied scenarios. *Mechatronics* **2020**, *65*, 102308. [CrossRef]
14. Lozano-Pérez, T.; Wesley, M.A. An algorithm for planning collision-free paths among polyhedral obstacles. *Commun. ACM* **1979**, *22*, 560–570 [CrossRef]
15. Lewis, F.L.; Ge, S.S. *Autonomous Mobile Robots: Sensing, Control, Decision Making and Applications*; CRC Press: Boca Raton, FL, USA, 2006.
16. Siegwart, R.; Nourbakhsh, I. R.; Scaramuzza, D. *Introduction to Autonomous Mobile Robots*; MIT Press: Cambridge, MA, USA, 2011.
17. Kagan, E.; Shvalb, N.; Ben-Gal, I. (Eds.) *Autonomous Mobile Robots and Multi-Robot Systems: Motion-Planning, Communication and Swarmingauthor*; John Wiley & Sons: New York, NY, USA, 2019.
18. LaValle, S.M. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*; TR 98-11; Computer Science Dept., Iowa State University: Ames, IA, USA, 1998
19. Véras, L.G.D.; Medeiros, F.L.; Guimaráes, L.N. Systematic Literature Review of Sampling Process in Rapidly-Exploring Random Trees. *IEEE Access* **2019**, *7*, 50933–50953. [CrossRef]
20. Chen, L.; Jafarnia-Jahromi, M.; Jain, R.; Luo, H. Implicit finite-horizon approximation and efficient optimal algorithms for stochastic shortest path. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 10849–10861.
21. Islam, F.; Nasir, J.; Malik, U.; Ayaz, Y.; Hasan, O. RRT star-Smart: Rapid convergence implementation of RRT star towards optimal solution. In Proceedings of the 2012 IEEE International Conference on Mechatronics and Automation, Chengdu, China, 5–8 August 2012; pp. 1651–1656.
22. Miao, C.; Chen, G.; Yan, C.; Wu, Y. Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm. *Comput. Ind. Eng.* **2021**, *156*, 107230. [CrossRef]
23. Agarwal, D.; Bharti, P.S. Implementing modified swarm intelligence algorithm based on Slime moulds for path planning and obstacle avoidance problem in mobile robots. *Appl. Soft Comput.* **2021**, *107*, 107372. [CrossRef]
24. Karur, K.; Sharma, N.; Dharmatti, C.; Siegel, J.E. A survey of path planning algorithms for mobile robots. *Vehicles* **2021**, *3*, 448–468. [CrossRef]

25. Zafar, M.N.; Mohanta, J.C.; Keshari, A. GWO-Potential Field Method for Mobile Robot Path Planning and Navigation Control. *Arabian J. Sci. Eng.* **2021**, *46*, 8087–8104. [CrossRef]
26. Sang, H.; You, Y.; Sun, X.; Zhou, Y.; Liu, F. The hybrid path planning algorithm based on improved A* and artificial potential field for unmanned surface vehicle formations. *Ocean Eng.* **2021**, *223*, 108709. [CrossRef]
27. Bałazy, P.; Gut, P.;Knap, P. Positioning algorithm for AGV autonomous driving platform based on artificial neural networks. *Robot. Syst. Appl.* **2021**, *1*, 41–45. [CrossRef]
28. Góra, K.; Kujawinski, M.; Wroński, D.; Granosik, G. Comparison of Energy Prediction Algorithms for Differential and Skid-Steer Drive Mobile Robots on Different Ground Surfaces. *Energies* **2021**, *14*, 6722. [CrossRef]
29. Hou, L.; Zhang, L.; Kim, J. Energy Modeling and Power Measurement for Mobile Robots. *Energies* **2019**, *12*, 19. [CrossRef]
30. Kim, C.; Kim, B. Energy-Saving 3-Step Velocity Control Algorithm for Battery-Powered Wheeled Mobile Robots. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain, 18–22 April 2005; pp. 2375–2380.
31. Berenz, V.; Tanaka, F.; Suzuki, K. Autonomous battery management for mobile robots based on risk and gain assessment. *Artif. Intell. Rev.* **2012**, 240, 217–237. [CrossRef]
32. Zhao, Y.; Rajeev, K.G.; Edeh, M.O. Robot visual navigation estimation and target localization based on neural network., Paladyn. *J. Behav. Robot.* **2022**, *13*, 76–83. [CrossRef]
33. Liu, S.; Sun, D. Modeling and experimental study for minimization of energy consumption of a mobile robot. In Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics, Kaohsiung, Taiwan, 11–14 July 2012; pp. 708–713.
34. Liu, S.; Sun, D. Minimizing Energy Consumption of Wheeled Mobile Robots via Optimal Motion Planning. *IEEE/ASME Trans. Mechatron.* **2014**, *19*, 401–411. [CrossRef]
35. Chellal, A.A.; Lima, J.; Gonçalves, J.; Megnafi, H. Battery Management System for Mobile Robots based on an Extended Kalman Filter Approch. In Proceedings of the 2021 29th Mediterranean Conference on Control and Automation (MED), Bari, Italy, 22–25 June 2021; pp. 1131–1136.
36. Tripathy, H.K.; Mishra, S.; Thakkar, H.K.; Rai, D. CARE: A collision-aware mobile robot navigation in grid environment using improved breadth first search. *Comput. Electr. Eng.* **2021**, *94*, 107327. [CrossRef]
37. Dudzik, S. Application of the Motion Capture System to Estimate the Accuracy of a Wheeled Mobile Robot Localization. *Energies* **2020**, *13*, 6437. [CrossRef]
38. QBot 2e. Available online: https://www.quanser.com/products/qbot-2e/ (accessed on 16 November 2022).
39. Autonomous Vehicles Research Studio. Available online: https://www.quanser.com/products/autonomous-vehicles-research-studio/ (accessed on 16 November 2022).