ECOLE CENTRALE DE LILLE

RESEARCH PROJECT G3

CAIO BRAGA SAMPAIO

Smart grid Integration of Self Driving Vehicles in urban networks

Supervisors:

M. Bruno FRANÇOIS

M. Alexandre KRUSZEWSKI

Villeneuve d'Ascq

2022

ACKNOWLEDGEMENT

I would like to recognize the support and motivation given by professor Bruno FRANÇOIS, who I had the pleasure to know since my arrival in France, showing a deep understanding to the difficulties of international students and trying to help them to develop in the best possible way.

Another special mention to professor Alexandre KRUSZESWKI, who accompanied me during the Track: Embedded Systems and Cyberphysics at Centrale Lille, which represented an arduous phase of my life, but of great learning and support from him.

Besides them, I'd like to thank my brazilian exchange colleagues who helped me a lot in the technical development of this work, especially my flatmates, Andreis Purim, João Lucas and Joel Kalil.

And finally, my family, who even far away, supported me and gave me all the necessary emotional support throughout the exchange.

ABSTRACT

This work presents studies on path-following self-driving ("autonomous") vehicles. The idea of the project arises from the problem of autonomous electric vehicles (AEVs) interconnected to smart grid systems, seeking charging stations in a way that does not prejudice the distribution network, avoiding problems such as exceeding maximum capacity or under voltage, for example.

Therefore, the proposal of this work constitutes the starting point for the development of this problematic. Thus, two main phases were developed: The designing of a path following controller for a robot, in a reduced scale, and a path calculation algorithm.

The control strategy adopted is composed of two cascaded loops, an internal loop with speed control of the robot motors, using a PI (Proportional-Integral) controller, and an external loop with position control of the robot, using a non-linear controller.

The algorithm for calculating the trajectory was based on the famous Dijkstra method, where given a graph of nodes, positions and links between them, the shortest path between a start point and an end point to be defined is calculated.

The implementation of this autonomous robot was performed with the mechanical structure of the Makeblock educational kit, Ultimate 10-in-1 Robot. Moreover, the systems embedded in the structure, used for the implementation of the control, counted with a MegaPi, control board based on Arduino MEGA 2560, and a Raspberry Pi 3 card, besides a MarvelMind kit for robot localization. The whole list of materials will be described later.

Keywords: Mobile Robots, Dijkstra's Algorithm,Trajectory Control

TABLE OF CONTENTS

1. INTRODUCTION

In recent decades the means of transport have undergone major changes and evolutions. Certainly, technology advances, and large mobility companies seek to adapt their products to better performance and innovative services, in the quest to attract new customers. However, global pressures for sustainable development have driven the development of greener and more efficient technologies, such as electric vehicles.

However, the introduction of these means of transportation implies new problems, such as the increase of load demand, which can significantly affect the distribution network, increasing peak power, changing the useful life of transformers and generating voltage variations [1].

Following this problematic, intelligent strategies of supervision and control of distribution networks are increasingly adopted, characterizing the smart grids. In this way, it is possible to identify points not so demanded by the power grid, which the charging of electric vehicles would not be so impactful. Thus, indicating these points to an autonomous electric vehicles (AEV), it could follow a trajectory to the nearest convenient charging point.

And the proposal of this work is the simulation of this situation in a reduced scale, in other words, indicate final points of the trajectory of a mini robot and implement a system to control its position.

The report is divided into 6 chapters, including this introduction. The topics covered in the following chapters will be:

Chapter 2: System specification

In this section the modelling of the robot dynamic system is presented.

Chapter 3: Design and analysis

In this section the robot control strategies are presented, that is, the control of the position and speed of the motors.

Chapter 4: Path Planning

In this section the algorithm for calculating the robot's trajectory will be presented, which seeks to find the shortest path between a series of pre-defined nodes (Dijkstra).

Chapter 5: Experimental Work

In this penultimate section, a review of the application and implementation of the studies done in the previous chapters will be given.

Chapter 6: Conclusion

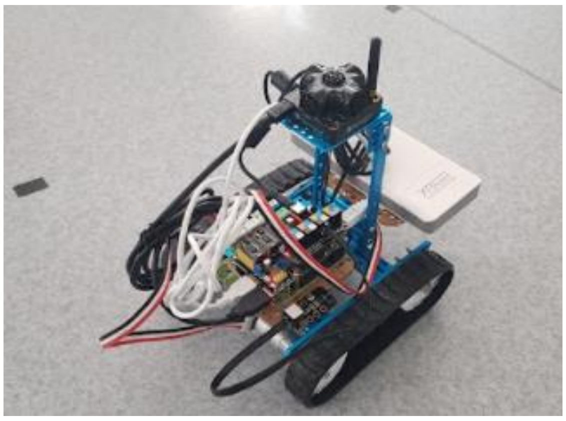
Finally, the conclusions obtained throughout the work are presented.

2. SYSTEM SPECIFICATION

2.1. Mechanical structure material used

Firstly, the mechanical system used for the studies of the work will be described.

Figure 1: Mini-robot used



Source: Author

The Ultimate 2.0 kit of the Makebloc platform, Figure 1, was used for the mechanical design of the robot. This kit allows the user a certain freedom of assembly, depending on its application.

It was chosen a tank type robot, adapted for the proposal of a trajectory-following robot and which allows the coupling of other accessories, such as the MarvelMind beacons for localization.

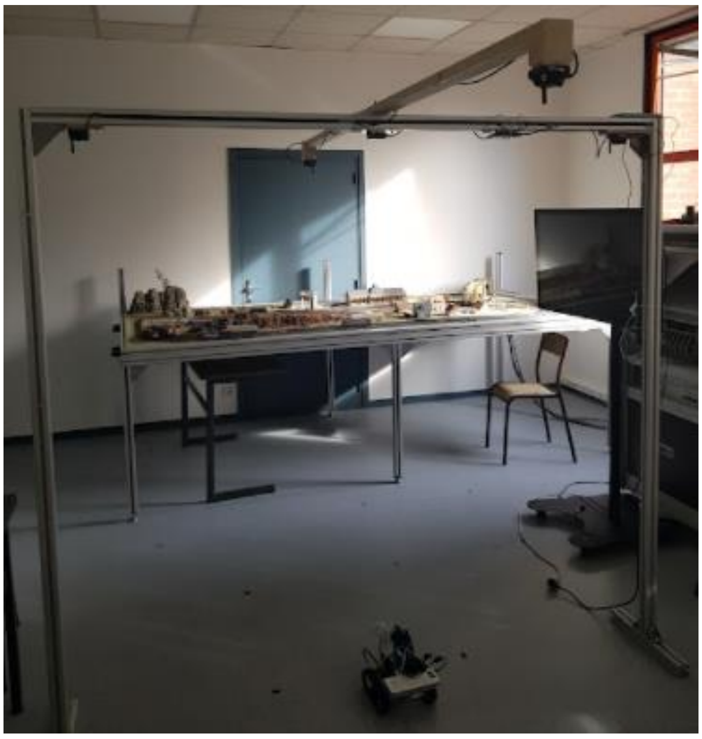
Among the materials, constituents of the system hardware, we can mention:

- A MegaPi control board, belonging to the Makebloc kit, based on the Arduino MEGA 2560.
- A Gyroscope sensor, for monitoring along the robot movement of its orientation.
- A Raspberry Pi 3 card.
- And a MarvelMind kit, which is an indoor positioning system similar to a GPS.

2.1.1. MarvelMind

The robot's position tracking strategy was done using the trilateration system proposed by the MarvelMind kit, the same logic as a GPS.

Figure 2: Implementation of the positioning system



Source: Author

The implementation of this system is done by installing beacons in certain points, example in Figure 2, of an indoor location, which remain fixed, preferably at higher heights or near the ceiling. And so, by means of an ultrasonic signal, it is possible to locate a mobile beacon, attached to the robot, identifying its coordinates in reference to the fixed beacons. More details of the application of this system are available in [2].

2.2. Modelling

Based on the structure to be used, in this section it will be possible to identify the mechanical model of the robot and equate its dynamics.

2.2.1. Identification of the model

Non holonomic constraint

Initially, it is possible to assume that the studied wheeled mobile robot (WMR) is a structure made of rigid elements, that is, its wheels, or tracks, are indeformable with respect to the horizontal plane [3].

Furthermore, the system to be applied follows non-holonomic constraints. This means that the algebraic relations depend on the speed and position of the vehicle. The constraint in this case, arises, from the mathematical expression of the conditions of wheel contact with the ground:

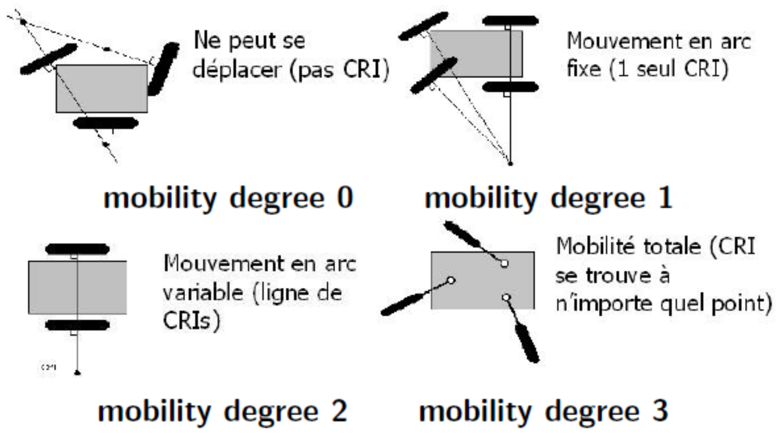
$$V = 0 = \dot{y}.\cos\theta - \dot{x}.\sin\theta \tag{1}$$

This implies that even if the robot returns to the initial configuration of the system, this does not guarantee the return of the system to the initial position, which does not occur in holomics systems.

Mobility (δm) and steering (δs) degrees

It is defined as mobility degree, the number of degrees of freedom for the movement of a robot. It is possible to identify from Figure 3 that the structure to be used has degree 2.

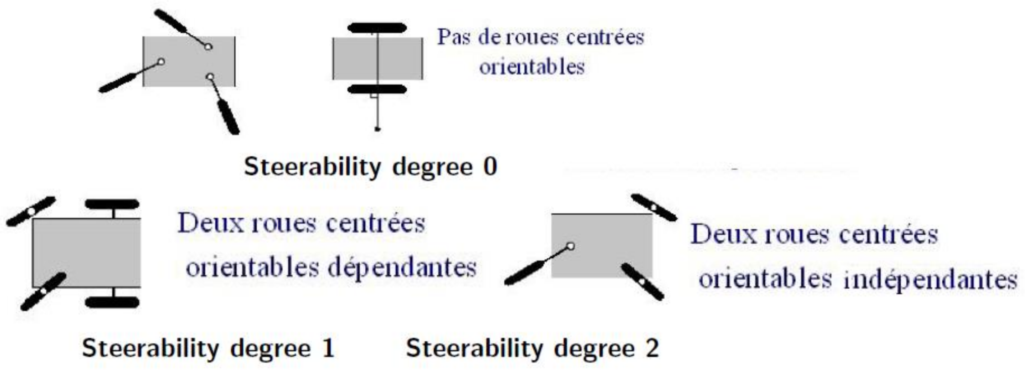
Figure 3: Types of mobility degree


Ne peut se déplacer (pas CRI)
mobility degree 0
Mouvement en arc fixe (1 seul CRI)
mobility degree 1
Mouvement en arc variable (ligne de CRIs)
mobility degree 2
Mobilité totale (CRI se trouve à n'importe quel point)
mobility degree 3

Source: [3]

The steerability degree is the number of centered wheels independent of the robot. Without this freedom in any of the wheels of the applied structure, we identify degree 0.

Figure 4: Types of steerability degree


Pas de roues centrées orientables
Steerability degree 0
Deux roues centrées orientables dépendantes
Deux roues centrées orientables indépendantes
Steerability degree 1
Steerability degree 2

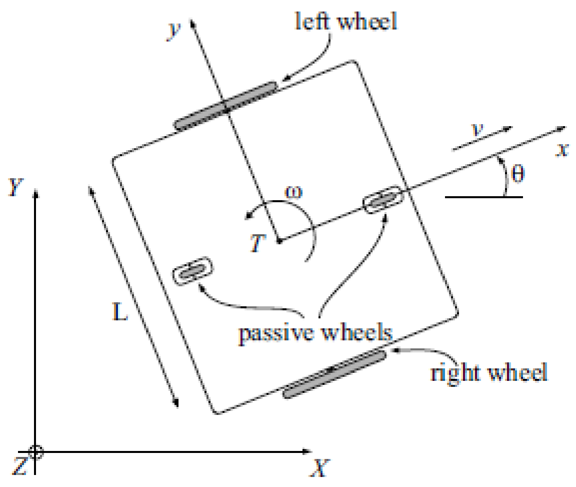
Source: [3]

2.2.2. Equationing the dynamic system

In terms of equating the system, it is also necessary to define the concept of posture. Which means a global description of the mobile robot in terms of its position. In most cases the posture of a robot is given by (x,y,ϴ). Where, x and y are its coordinates in the cartesian axes and ϴ its angular orientation with respect to the x-axis.

Thus, it is possible to identify the dynamic equations of this type of robot, δm = 2 and δs = 0, identified in Figure 5.

Figure 5: Robot architecture and symbols



Source: [4]

Where the linear and angular velocities of the robot are represented by v and ω, respectively.

$$\begin{aligned} \dot{x} &= v.\cos\theta \\ \dot{y} &= v.sin\,\theta \\ \dot{\theta} &= \omega \end{aligned} \tag{2}$$

Or even,

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} . \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{3}$$

Another interesting data that can be extracted from the dynamic relations of the robot, are the equations between the linear and angular velocities of the robot with the individual linear velocities of each wheel. [4]

That is, it can be agreed that the velocity of the left wheel (vl) is given by:

$$vl = v - \frac{\omega.L}{2} \tag{4}$$

And the speed of the right hand wheel (vr) is given by:

$$vr = v + \frac{\omega.L}{2} \tag{5}$$

Isolating v and ω,

$$v = \frac{(vl + vd)}{2}$$

$$\omega = \frac{(vd - vl)}{L} \tag{6}$$

Where, L is the robot axis width, in the case of this project has value equal 20 cm.

3. DESIGN AND ANALYSIS

In this section, the adopted control strategy will be introduced, which is divided into two main parts, the individual speed control of each wheel of the robot and the position control.

3.1. Motor Speed Control

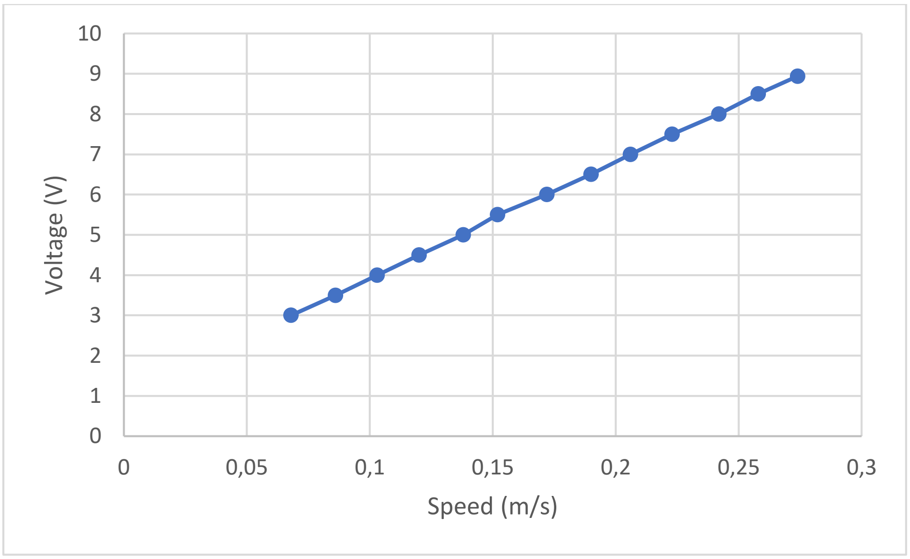
For the speed control of the motors, two tests were previously performed, which were intended to identify the dynamics of the motor drive system.

3.1.1. Test for checking the voltage and speed relation of the motors

Initially, in order to facilitate the implementation of the control code in the MegaPi board, the relationship between tension and speed of each wheel was studied.

In a short test with MegaPi and the motors, increasing voltages were applied to the motors and, based on the encoder data, their linear speed was analysed. The result of this test is illustrated below.

Figure 6: Linearity between voltage and speed



Source: Author

What can be concluded from this test is that the relationship between applied voltage and motor speed follows a linear tendency, represented by the following equation.

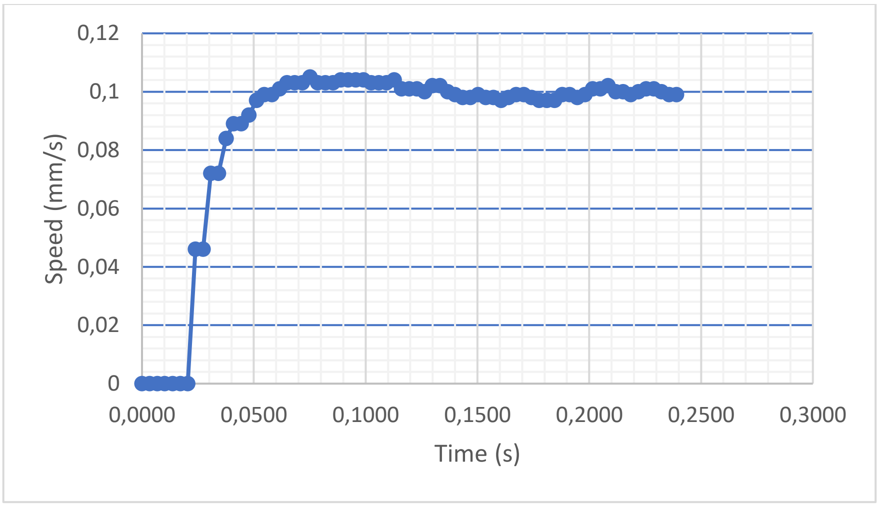
$$U = 28.83.v + 1.04 \tag{7}$$

Where U is the applied voltage, in volts, and v is the linear velocity of the wheel, in m/s.

3.1.2. Step response

Subsequently, a second test was performed, now with the intention of identifying the dynamic motor drive system, verifying its response to a step. In this case, as the voltage-speed relationship is already known, 0.1 m/s was applied as equivalent to a step input during the test. The result of this, is illustrated in the following figure.

Figure 7: Test of the step response of the motors



Source: Author

Considering that the resulting system follows a first-order dynamics, it is possible to find its transfer function as follows.

$$G(s) = \frac{K}{\tau.s + 1} \tag{8}$$

Having applied a voltage of 0.1 volts to the motor, and having as a result in the permanent regime the same voltage value, it is possible to conclude that the gain K is unitary.

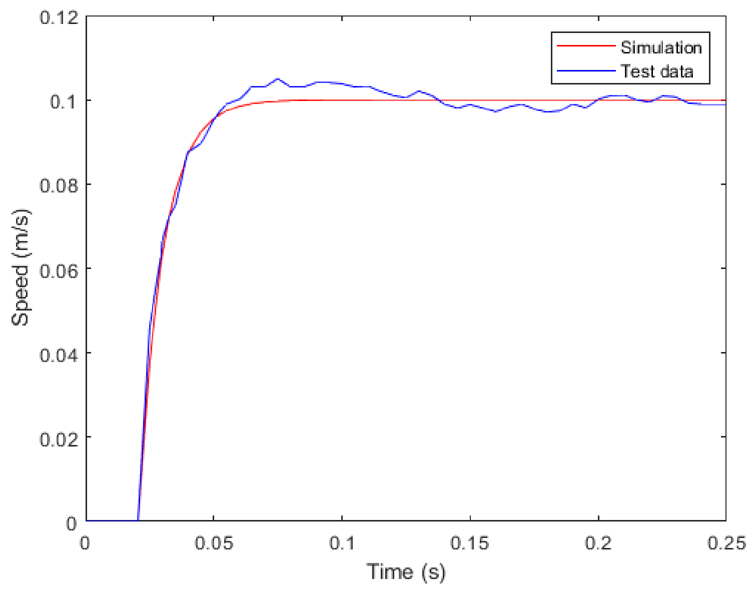
For the value of t (tau), the ratio of rise time up to 63% of the final value is known, as the value of this time constant. Thus, it is verified that in this case, the system takes 30 ms to reach 0.063 volts. Taking into account the delay of 20.5 ms, t(tau) is estimated as 9.5 ms.

It is then concluded that the transfer function of the motor drive system is:

$$G(s) = \frac{1}{0.0095.s + 1} \tag{9}$$

Finally, in terms of comparison, the data from the previous test was plotted, along with the simulation of the step response (0.1 m/s) of the transfer function obtained in (9).

Figure 8: Comparison of simulation and test of step response of the motors



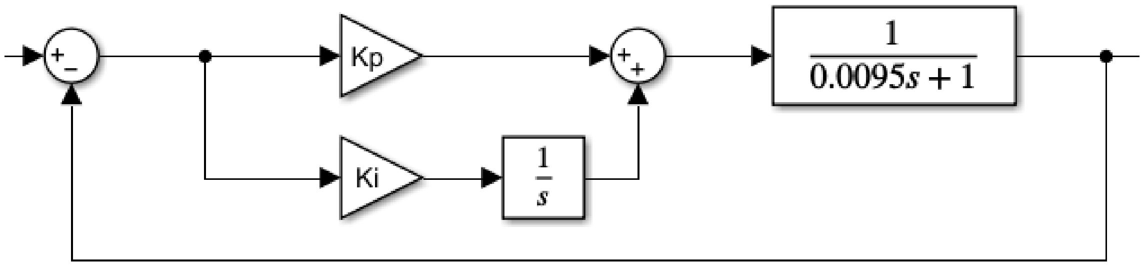
Source: Author

Thus, it is possible to validate the transfer function, considering its similarity with the curve obtained from the experimental test data.

3.1.3. PI controller

The speed control strategy of the motors is based on the use of a Proportional Integrator (PI) controller, whose scheme is illustrated in the following figure.

Figure 9: Closed loop wheel speed control



Source: Author

Once the transfer functions are known after the addition of the controller, it is possible to adjust the gains, according to the needs foreseen for the project. Thus, following [5], we have that:

$$G(s) = \frac{K.(Kp.s + Ki)}{K.(Kp.s + Ki) + s(\tau.s + 1)}$$

$$G(s) = \frac{\frac{K.Kp}{\tau}.s + \frac{K.Ki}{\tau}}{s^2 + \frac{(K.Kp + 1)}{\tau}.s + \frac{K.Ki}{\tau}} \qquad (10)$$

So, it is notorious that the addition of the PI controller, results in a second order system. Which allows us to conclude that:

$$Ki = \frac{\tau.\omega^2}{K}$$

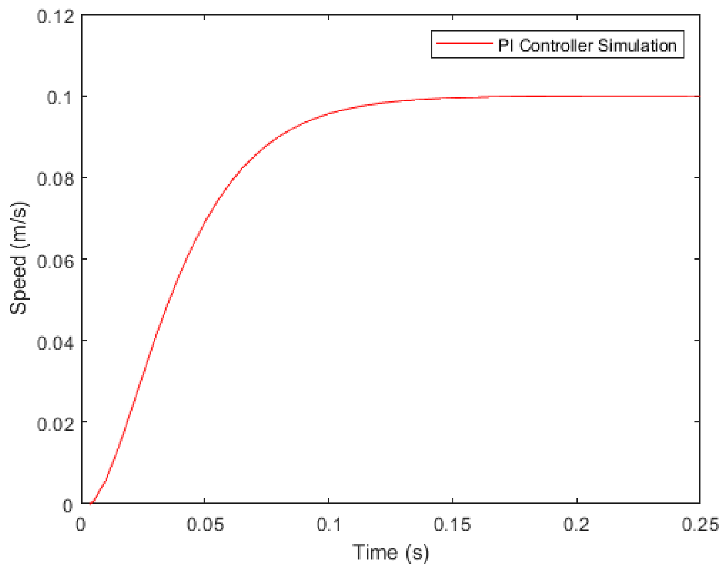
$$Kp = \frac{(2.\xi.\tau.\omega - 1)}{K} \qquad (11)$$

By the properties of second order systems, it is convenient for the design to use a damping coefficient ξ unitary, because such a value allows the system to be critically damped. Which allows the calculation of ω, for the desired response time, as follows:

$$\omega = \frac{5}{Tr} \tag{12}$$

A response time equal to 0.1 seconds is adopted, which results in an ω equal to 50. As a consequence, Ki = 23.75 and Kp = -0.05. Which results in the performance illustrated in Figure 10.

Figure 10: PI controller simulation



Source: Author

3.2. Path-following controller

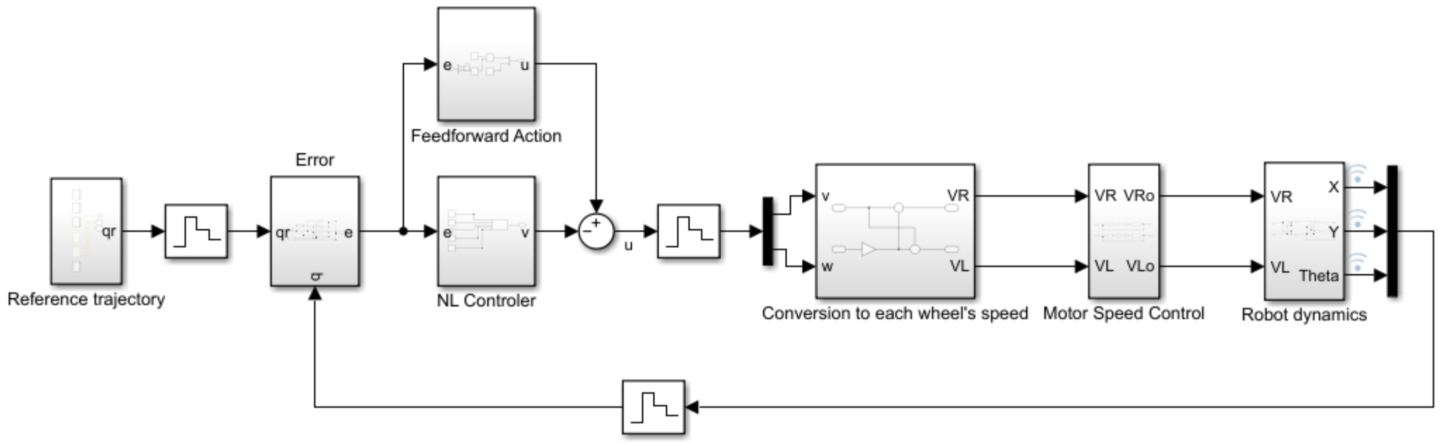
3.2.1. Controller parameters

Finally, the position control strategy, which was based on a non-linear controller from the study of [4] and [6], taking into account the possibility of making large angular variations.

The objectives of this controller are to indicate points of a predefined trajectory and ensure that the robot follows these points.

The block diagram of this control is shown in figure 11, and each block will be explained in sequence.

Figure 11: Schematic of position control



Source: Author

Reference trajectory:

It is necessary to indicate for each point of the path to be followed by the robot, a reference posture qr(xr,yr,Ɵr), that is, the Cartesian positions of the points and the angles that the robot must reach, taking its current position q(x,y,Ɵ) as a basis.

Error:

Initially, the robot's posture error with respect to the desired path is determined. Normally, the error is calculated only by the difference of a certain reference value and the current value. However, in this case, it is necessary to consider possible rotation of the robot, and so we take into account the following general form of the error:

$$\begin{bmatrix} e1 \\ e2 \\ e3 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} xr - x \\ yr - y \\ \theta r - \theta \end{bmatrix} \tag{13}$$

FeedFoward Action:

Deriving the above relationship from the error, and the kinematic model illustrated in (3), the following relationship is found:

$$\begin{bmatrix} \dot{e}1 \\ \dot{e}2 \\ \dot{e}3 \end{bmatrix} = \begin{bmatrix} \cos e3 & 0 \\ \sin e3 & 0 \\ 0 & 1 \end{bmatrix} . \begin{bmatrix} ur1 \\ ur2 \end{bmatrix} + \begin{bmatrix} -1 & e2 \\ 0 & -e1 \\ 0 & -1 \end{bmatrix} . \begin{bmatrix} u1 \\ u2 \end{bmatrix} \tag{14}$$

Where ur1 is the tangential feedfoward speed and ur2 is the angular feedfoward speed. The robot inputs, u1 and u2, under relation (14) can be expressed as follows:

$$u1 = ur1.cose3 - v1$$

$$u2 = ur2 - v2$$

(15)

Where v1 and v2, are the outputs of the non-linear controller.

Non Linear Controller:

And finally, the controller is defined as:

$$\begin{bmatrix} v1 \\ v2 \end{bmatrix} = \begin{bmatrix} -k1 & 0 & 0 \\ 0 & -sign(ur1).k2 & -k3 \end{bmatrix} . \begin{bmatrix} e1 \\ e2 \\ e3 \end{bmatrix} \tag{16}$$

Where gains are calculated according to the reference trajectory:

$$k1 = k3 = 2.\xi.\omega n(t)$$

$$k2 = g.|ur1(t)| \tag{17}$$

And the values of ξ and g are set by the user in order to adjust the desired response:

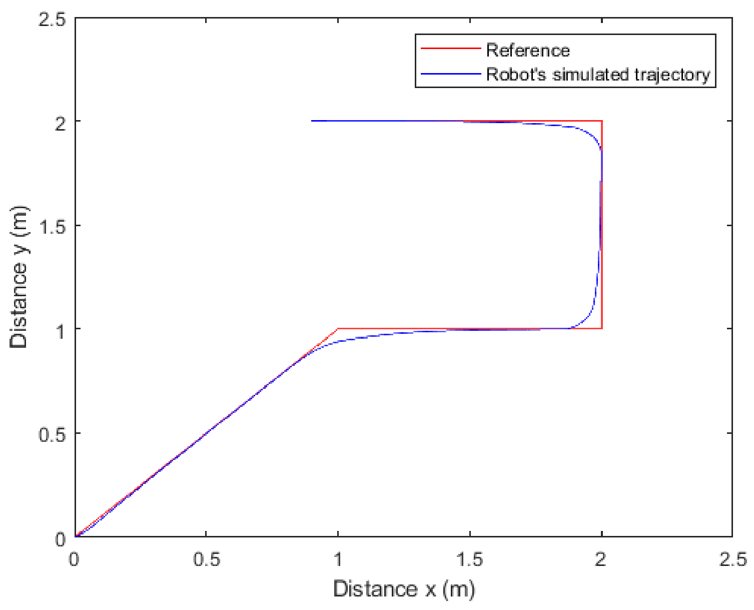
$$\omega n(t) = \sqrt{ur2^2(t) + g.ur1^2(t)} \tag{18}$$

3.2.2. Simulation results

Thus, the controller was analysed with the parameters recommended in [Klancar, Matko, & Blazic, 2005], ξ = 0.9, g=30 and a reference trajectory was considered, starting from the point (0,0,0), following the following postures: [(1.0,1.0,π/4), (2.0,1.0,0), (2.0,2.0,π/2), (1.0,2.0,π)], with the units in meters for the Cartesian points and in radians for the desired orientation.

In addition, it was used as reference linear velocity (ur1) equal to 0.1 m/s and as reference angular velocity (ur2) equal to 0 rad/s. The simulation result of this trajectory is illustrated in figure 12.

Figure 12: Simulation of trajectory control



Source: Author

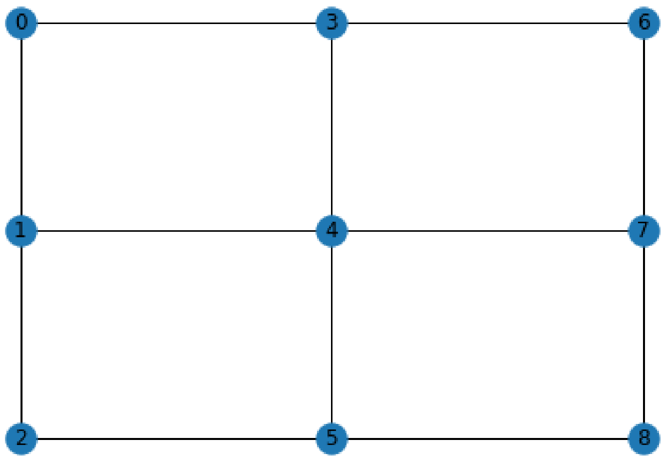
It is noted that the results are satisfactory and the robot follows, adequately, the reference trajectory. The total simulation time was 90 seconds, updating the new point of the trajectory every 20 seconds. Besides, a sampling time of 0.1 seconds was used.

4. PATH PLANNING

To calculate the trajectory, the famous Dijkstra’s algorithm was used, which allows finding the shortest path between two points, given a graph.

By definition, a graph is nothing more than a set of nodes and the indication of the connection between them. Figure 13 illustrates a graph that could be applied to calculate the robot's trajectory. For its implementation, for instance, it would be enough to identify the coordinates of each of the nine nodes, by means of the MarvelMind positioning. In the experimental tests, the corner nodes (0, 2, 6 and 8) were identified as the fixed beacons of the MarvelMind kit, since they were installed in a square shape like the one represented by the graph.

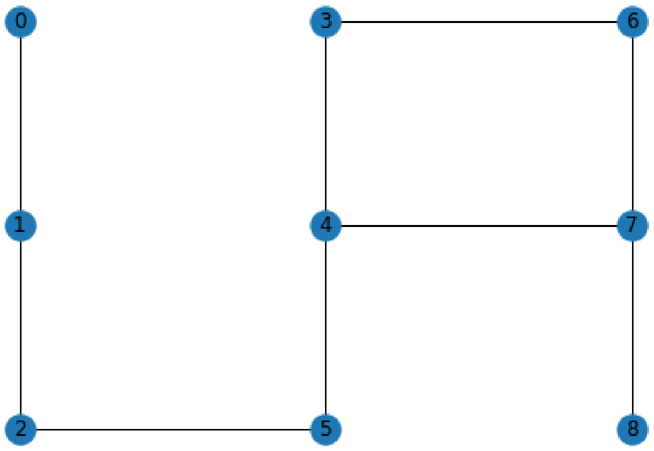
Figure 13: Example of a graph with all nodes connected



Source: Author

In this graph, it is possible to see that all nodes are interconnected and plausible to compose a path. However, not all nodes need to be interconnected, in practice, restrictions can be imposed, such as obstacles for example, preventing the connection of some nodes, which is polished in figure 14.

Figure 14: Example of a graph with not all nodes connected



Source: Author

In simplified form the algorithm works as follows, [7]:

(i) Identify the desired end point and consider it with a zero distance and current point.

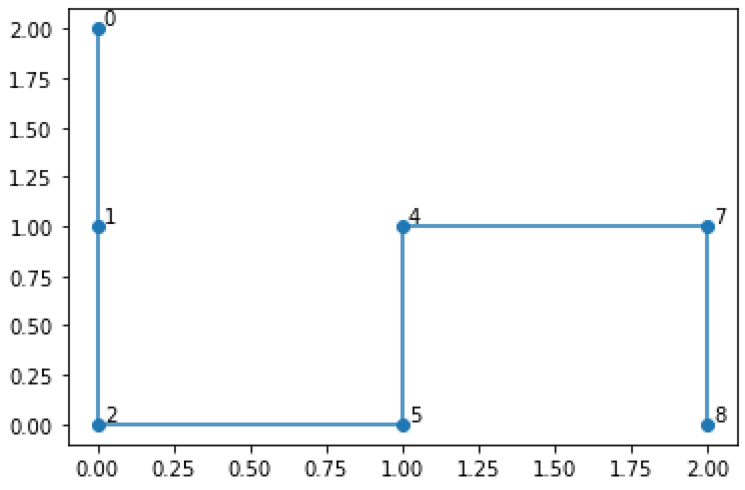
(ii) Find all the points connected to the current point. Check their distances and record the smallest

(iii) Mark the current point as visited and look no further at it

(iv) Mark the shortest distance point as current and repeat from step (ii)

This way, using the graph of figure 14, the shortest path was calculated, considering the distances of a square of 2 meters side, as the area occupied by the nodes, resulting in the trajectory of figure 15.

Figure 15: Trajectory found by Dijkstra's algorithm



Source: Author

5. EXPERIMENTAL WORK

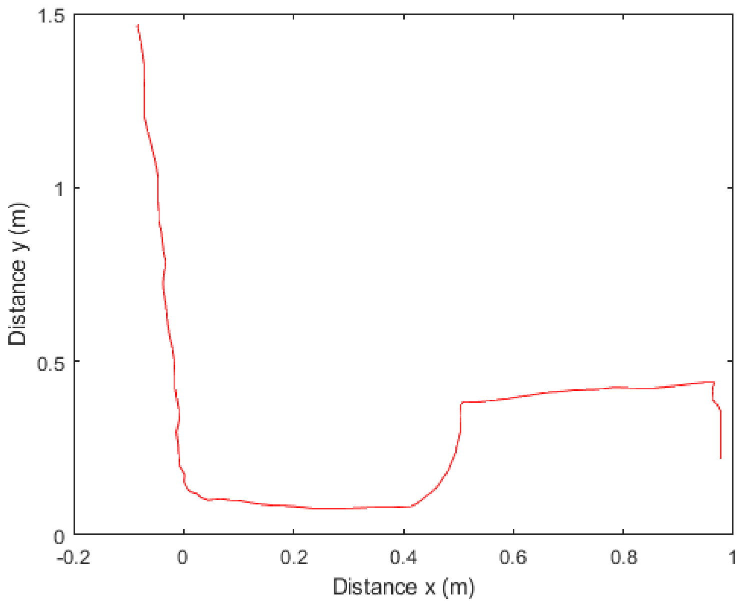
Finally, the robot trajectory control was implemented, along with Dijkstra's algorithm. By placing obstacles preventing the connection of nodes, equal to the situation of the graph of figure 14, the positioning of each node was adjusted, according to the circumstances of the room where the experiment was performed, resulting in the following trajectory, executed as expected.

The link to the video of the demonstration of this experiment is available below:

https://drive.google.com/file/d/1zmzxOGDYZDRcaSAGkBV0fvvwh0hxGi5u/view?usp=sharing

And the graphical results of the test are exposed in Figure 16.

Figure 16: Trajectory experimentally performed by the robot



Source: Author

6. CONCLUSION

In this work, the application of a complex system of monitoring, control and trajectory calculation was presented. The results obtained in simulation and experimental were satisfactory, even considering the imprecisions of sensors and localization systems.

As next steps in the development of this project, it is suggested the implementation of a control in a 4-wheeled robot, making it closer to reality. Besides, it would be interesting to implement the prototype in more extensive environments, allowing the robot to trace longer trajectories.

BIBLIOGRAPHY

[1] S. Habib, M. Kamran, and U. Rashid, “Impact analysis of vehicle-to-grid technology and charging strategies of electric vehicles on distribution networks–a review,” Journal of Power Sources, vol. 277, pp. 205–214, 2015.

[2] For Autonomous Vehicles, robots, drones ... - marvelmind. (n.d.), from https://www.marvelmind.com/pics/marvelmind_presentation.pdf

[3] Perruquetti, W. *Autonomous transport system: from modelling to path planning and control* (Chapter 5). VILLENEUVE DASCQ.

[4] G. Klancar, D. Matko and S. Blazic, "Mobile Robot Control on a Reference Path," Proceedings of the 2005 IEEE International Symposium on, Mediterrean Conference on Control and Automation Intelligent Control, 2005., 2005, pp. 1343-1348

[5] PID Control - Maplesoft. (n.d.). Retrieved March 29, 2022, from https://www.maplesoft.com/content/EngineeringFundamentals/12/MapleDocument_12/PID%20Control.pdf

[6] KANAYAMA, Y. A Stable Tracking Control Method for an Autonomous Mobile Robot. Proceedings of the IEEE Conference on Robotics Automation. Santa Bárbara CA, p. 384 - 389. 1990.

[7]Lippman, D. (n.d.). Mathematics for the liberal arts. Lumen. Retrieved March 30,

2022,from:https://courses.lumenlearning.com/waymakermath4libarts/chapter/shortest-path/