BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM
KHOA ĐÀO TẠO CHẤT LƯỢNG CAO

MECHATRONICS PROJECT REPORT
Major of MECHATRONIC ENGINEERING TECHNOLOGY

DEVELOPMENT OF A MOBILE SERVICE ROBOT

ADVISOR: M.Sc. Võ Lâm Chương

STUDENTS: Huỳnh Anh Duy - 20146031
Nguyễn Hải Hoàng - 20146004
Nguyễn Ngọc Nhân - 20146262

Hồ Chí Minh city, 26th May 2023

LIST OF PARTICIPATING MEMBERS

1ST SEMESTER, 2022-2023

NO.	Student's name	ID	Completion rate
1	Nguyễn Hải Hoàng	20146004	100%
2	Nguyễn Ngọc Nhân	20146262	100%
3	Huỳnh Anh Duy	20146031	100%

Note:

- Percentage % = 100%: Percentage level of each student participating.
- Team leader: Nguyễn Hải Hoàng

COMMENT

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……

Sign

M.Sc. Võ Lâm Chương

HCMC UNIVERSITY OF TECHNOLOGY AND EDUCATION

FACULTY FOR HIGH QUALITY TRAINING

THE SOCIALIST REPUBLIC OF VIETNAM

Independence - Freedom – Happiness

PROJECT OF MECHATRONICS SYSTEM ASSIGNMENT

1ST SEMESTER / Year 2023-2024

Advisor: M.Sc. Võ Lâm Chương

Student name: Nguyễn Hải Hoàng	ID: 20146004	Phone No.: 0868129935
Student name: Nguyễn Ngọc Nhân	ID: 20146262	Phone No.: 0353190443
Student name: Huỳnh Anh Duy	ID: 20146031	Phone No.: 0966908617

1. *Project title:* DEVELOPMENT OF A MOBILE SERVICE ROBOT

2. Initial materials provided by the advisor:

……

……

……

3.Content of the project:

……

……

4. Final product:

……

……

5. Date of assignment:

6. Date of submission: 06/01/2024

CHAIR OF THE PROGRAM
(Sign with full name)

ADVISOR
(Sign with full name)

☐ Approval for oral defense ……
(Advisor sign with full name)

HCMC UNIVERSITY OF TECHNOLOGY AND EDUCATION
FACULTY FOR HIGH QUALITY TRAINING

THE SOCIALIST REPUBLIC OF VIETNAM
Independence - Freedom – Happiness

ADVISOR'S EVALUATION SHEET

Student name: Nguyễn Hải Hoàng	ID: 20146004	Phone No.: 0868129935
Student name: Nguyễn Ngọc Nhân	ID: 20146262	Phone No.: 0353190443
Student name: Huỳnh Anh Duy	ID: 20146031	Phone No.: 0966908617

Project title: **DEVELOPMENT OF A MOBILE SERVICE ROBOT**

Major: Mechatronics engineering

Advisor: M.Sc. Võ Lâm Chương

EVALUATION:

1. Content of the project:

……
……
……

2. Strengths:

……
……
……

3. Weaknesses:

……
……
……

4. Approval for oral defense? *(Approved or denied)*

……

5. Overall evaluation: *(Excellent, Good, Fair, Poor)*

……..……..……..……

6. Mark …… (*in words*: ……..)

Ho Chi Minh City, Day Month Year

ADVISOR

(Sign with full name)

HCMC UNIVERSITY OF TECHNOLOGY AND EDUCATION
FACULTY FOR HIGH QUALITY TRAINING

THE SOCIALIST REPUBLIC OF VIETNAM
Independence - Freedom – Happiness

PRE-DEFENSE EVALUATION SHEET

Student name: Nguyễn Hải Hoàng	ID: 20146004	Phone No.: 0868129935
Student name: Nguyễn Ngọc Nhân	ID: 20146262	Phone No.: 0353190443
Student name: Huỳnh Anh Duy	ID: 20146031	Phone No.: 0966908617

Project title: **DEVELOPMENT OF A MOBILE SERVICE ROBOT**

Major: Mechatronics engineering

Name of Reviewer: M.Sc. Võ Lâm Chương

EVALUATION:

1. Content of the project:

……
……
……

2. Strengths:

……
……
……

3. Weaknesses:

……
……
……

4. Approval for oral defense? (*Approved or denied*)

……

5. Overall evaluation: (*Excellent, Good, Fair, Poor*)

……..……..……..……

6. Mark …… (*in words*: ……..)

Ho Chi Minh City, *Day* *Month* *Year*

REVIEWER

(Sign with full name)

HCMC UNIVERSITY OF TECHNOLOGY AND EDUCATION
FACULTY FOR HIGH QUALITY TRAINING

THE SOCIALIST REPUBLIC OF VIETNAM
Independence - Freedom – Happiness

EVALUATION SHEET OF DEFENSE COMMITTEE MEMBER

Student name: Nguyễn Hải Hoàng	ID: 20146004	Phone No.: 0868129935
Student name: Nguyễn Ngọc Nhân	ID: 20146262	Phone No.: 0353190443
Student name: Huỳnh Anh Duy	ID: 20146031	Phone No.: 0966908617

Project title: **DEVELOPMENT OF A MOBILE SERVICE ROBOT**

Major: Mechatronics engineering

Advisor: M.Sc. Võ Lâm Chương

Name of Defense Committee Member:

……..……..……..……...

EVALUATION:

1. Content and workload of the project:

……..……..……
……..……..……
……..……..……

2. Strengths:

……..……..……
……..……..……
……..……..……

3. Weaknesses:

……..……..……
……..……..……
……..……..……

4. Overall evaluation: *(Excellent, Good, Fair, Poor)*

……..……..……

5. Mark …… (*in words*: ……..)

Ho Chi Minh City, *Day Month Year*

COMMITTEE MEMBER

(Sign with full name)

Table of Content

List of figures

List of tables

1 INTRODUCTION

1.1 Motivation

The motivation behind the development of a mobile service robot to guide students and teachers to their destination classrooms in a school setting stems from the desire to enhance the educational experience and promote efficiency within the learning environment. Navigating through a large and complex school building can often be a challenge, particularly for newcomers or individuals with limited familiarity with the premises. This can lead to unnecessary stress, delays, and disruptions to the learning process.

By introducing a mobile service robot with advanced navigation capabilities, we aim to alleviate these challenges and create a seamless and user-friendly experience for students and teachers. The project seeks to leverage the power of robotics and automation to provide personalized guidance and support, ensuring that individuals can easily and efficiently find their way to their desired classrooms.

The project also aims to foster independence and self-reliance among students. By empowering them to navigate the school environment with the assistance of a mobile service robot, we encourage students to take ownership of their educational journey and develop essential skills in problem-solving, decision-making, and adaptability. Furthermore, the incorporation of robotics technology in the educational landscape serves as a catalyst for inspiring interest and engagement in science, technology, engineering, and mathematics (STEM) disciplines.

Fig. 1: *Robots aid homebound students as a stand in* [1]

Moreover, the project aligns with the broader trend of integrating smart technologies into various sectors to enhance efficiency and improve user experiences. By harnessing the capabilities of Robotino and implementing intelligent algorithms for mapping, localization, and path planning, we can optimize navigation routes, reduce congestion, and minimize the time spent searching for classrooms. This not only benefits students and teachers but also contributes to the overall productivity and smooth functioning of the educational institution.

Additionally, the project has the potential to serve as a prototype and foundation for future developments in the field of mobile service robotics. The knowledge and insights gained from

this project can be applied to other domains requiring navigation assistance, such as hospitals, shopping malls, and airports. By pushing the boundaries of robotic capabilities and exploring new possibilities, we contribute to the advancement of technology and its application in real-world contexts.

1.2 Background

1.2.1 History and State of the Service Robots

Service robots have made significant strides in enhancing school education by providing innovative learning experiences and support for students' development. The history of service robots in education traces back to the early 2000s, when educational robots like LEGO Mindstorms gained popularity, allowing students to build and program their own robots. As technology advanced, purpose-built service robots specifically designed for education entered the market. Today, service robots in school education have evolved to become more sophisticated and versatile. They play a pivotal role in STEM education, offering hands-on learning experiences in science, technology, engineering, and mathematics. These robots also facilitate personalized learning by tailoring instruction and activities based on individual student needs, fostering a more engaging and effective learning environment.
Service robots also promote coding and programming skills, offering built-in platforms for students to learn computational thinking and logical reasoning. They foster collaborative learning experiences by encouraging teamwork and cooperation among students. It is important to note that service robots are meant to complement human teachers, not replace them. They expand learning opportunities and provide additional support, but human interaction remains crucial in education. As service robots continue to advance, we can anticipate further integration in schools, providing even more engaging and interactive learning experiences for students. The sections below will provide an overview of significant innovations in the field, which are particularly relevant to the ongoing development of our own robot.

1.2.2 Starship autonomous delivery robot

Starship is a six-wheeled ground robot that can navigate streets and sidewalks autonomously. It's designed to be safe and robust, offering on-demand package delivery for consumers and businesses. Starship Technologies announced that it will begin delivery service on four additional college campuses this fall (2021): University of Illinois Chicago (UIC), University of Kentucky (UK), University of Nevada, Reno (UNR) and Embry-Riddle Aeronautical University's Daytona Beach, FL campus. These campuses join the list of college campuses across the US where Starship robots will continue to provide deliveries this fall with its global fleet of over 1,000 robots.

Fig. 2: *Students can order and receive deliveries conveniently from just about anywhere on campus.*

[2]

1.2.3 LG CLOI GUIDEBOT

The LG Cloi GuideBot is a multi-purpose customer service robot designed to revolutionize the customer experience. LG GuideBot can navigate busy spaces and provide facility guidance, destination accompaniment, signage advertising, security patrol, and photography services. It features a touch-interactive large 2-sided LCD display, speech recognition, and multi-language support, making it the ideal workplace companion for businesses looking to enhance their customer experience, increase efficiency, and streamline operations.

Fig. 3: *The Guidebot conveniently guides visitors to artworks.*
[3]

2 MECHANICAL DESIGN

2.1 Overview of Robotino mobile robot

The mechanical design of Festo Robotino plays a crucial role in its functionality and versatility as a mobile robot platform. The robot's design combines robustness and agility to enable smooth movement and reliable operation in various environments. At its core, Robotino features a sturdy chassis with a well-engineered layout that accommodates its essential components. The platform is equipped with three wheels, strategically positioned to provide omnidirectional movement capabilities. This design allows Robotino to navigate tight spaces, make precise turns, and move in any direction with ease. The mechanical design also incorporates sensors and actuators essential for perception and interaction with the environment. Proximity sensors and distance measurement sensors enable the robot to detect and avoid obstacles, ensuring safe and efficient navigation. Furthermore, a laser scanner is included, enabling mapping and localization capabilities for accurate positioning. The overall design of Robotino focuses on durability, stability, and adaptability, making it suitable for a wide range of applications. Its mechanical design not only facilitates effective operation but also enhances the user experience by providing a robust and reliable platform for educational and industrial purposes.

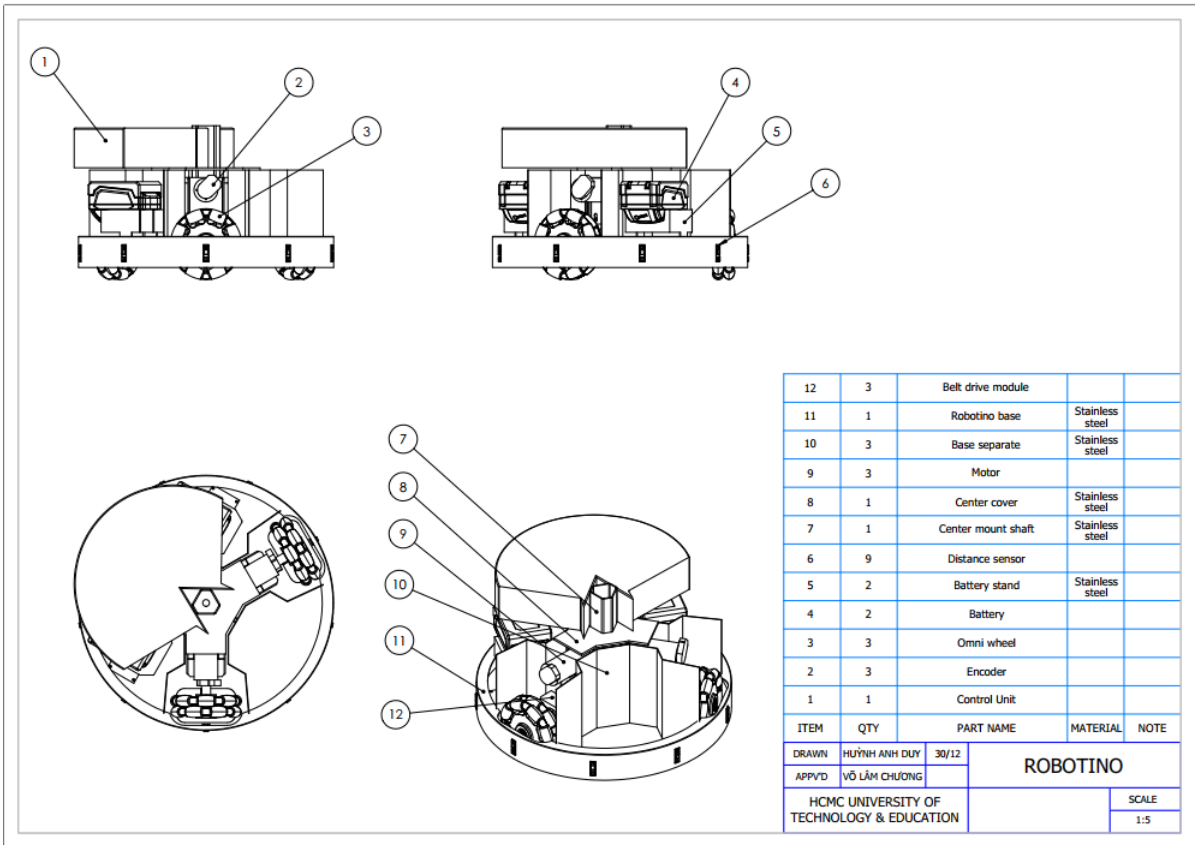


Fig. 4: *Overview of Robotino mobile robot.*

Parameter	Value
Height	325 mm
Diameter	450 mm
Total weight (curb weight)	20 kg
Total weight (including 4 battery packs)	22,8 kg (ca. 700 g each accumulator)
Maximum payload	30 kg (centered)
IP Protection	IP 00
Battery voltage	18 V
Housing material	stainless steel, PA6
Degrees of freedom	3 (translatory in the x and y direction, rotational about the z axis)

Table 1: *General specifications of Robotino*

2.2 Selection for transmission method

The belt transmission method has been chosen for the motor-to-gearbox power transfer in this project. Belt drives offer several advantages that make them a suitable choice for this application. Firstly, they provide smooth and quiet operation, minimizing noise and vibration levels during operation. This is especially important in a school environment where a quiet and peaceful atmosphere is desired. Additionally, belt drives are known for their high efficiency, allowing for effective power transfer with minimal energy loss. This ensures optimal performance and reduces energy consumption, making it a cost-effective solution.
Another advantage of belt transmission is its ability to absorb shock and dampen vibrations. This is particularly beneficial in a mobile service robot where the robot may encounter uneven surfaces or obstacles during navigation. The flexibility of the belt allows it to absorb shocks and vibrations, protecting both the motor and gearbox from potential damage. Moreover, belt drives are relatively easy to install and maintain. They do not require lubrication and are less prone to wear and tear compared to other power transmission methods. In the event of belt failure, replacement is relatively straightforward, minimizing downtime and ensuring the robot can resume its operations quickly.

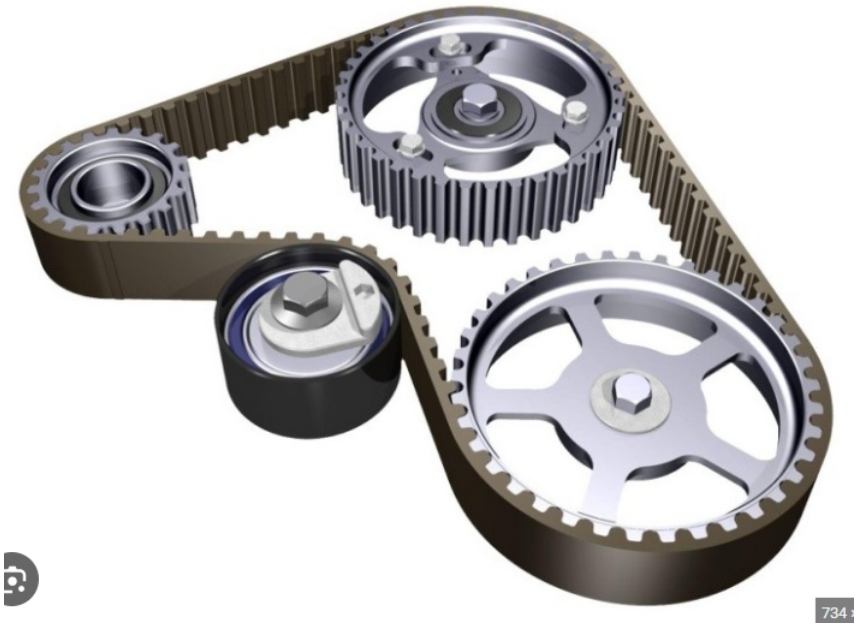


Fig. 5: *Belt drive transmission.*
[4]

Therefore, in Robotino's one transmission assembly, a DC motor drives through a gearbox using a belt transmission, with a GT2 pulley - 15 teeth on the motor shaft and a GT2 pulley - 60 teeth on the gearbox shaft, to achieve a transmission ratio of 1:4.

2.3 Belt drive module design

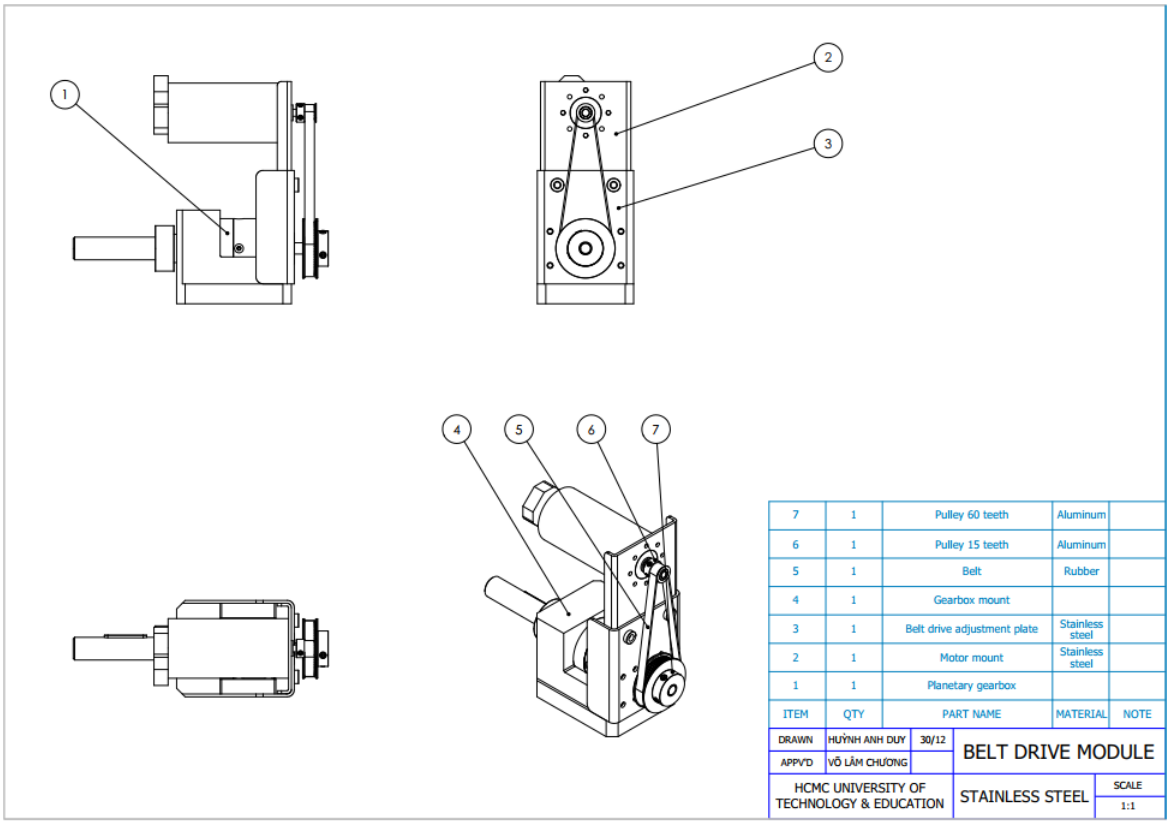


Fig. 6: *Belt drive module in Robotino.*

In a transmission system of Robotino, designed by the manufacturer, a DC motor drives through a planetary gearbox (with a gear ratio of 1:8) using a belt transmission. The gearbox and motor are securely fixed using aluminum and stainless steel brackets, allowing for adjustable belt tension through the design of elongated screw slots. The total ratio result in 1:32.

No.	**Part name**	**Material**
1	Planetary gearbox	
2	Motor mount	Stainless steel
3	Belt drive adjustment plate	Stainless steel
4	Gearbox mount	Aluminum
5	Belt	Rubber
6	Pulley 15 teeth	Aluminum
7	Pulley 60 teeth	Aluminum

Table 2: *Parts list in one belt drive module.*

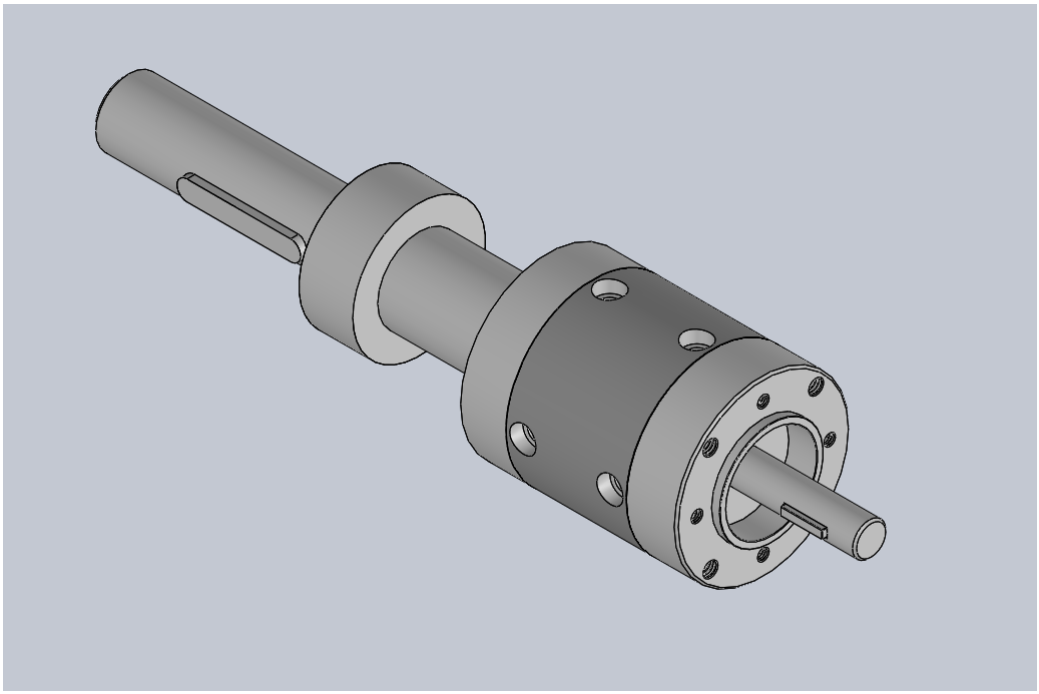


Fig. 7: *Planetary gearbox - Ratio 1:8.*

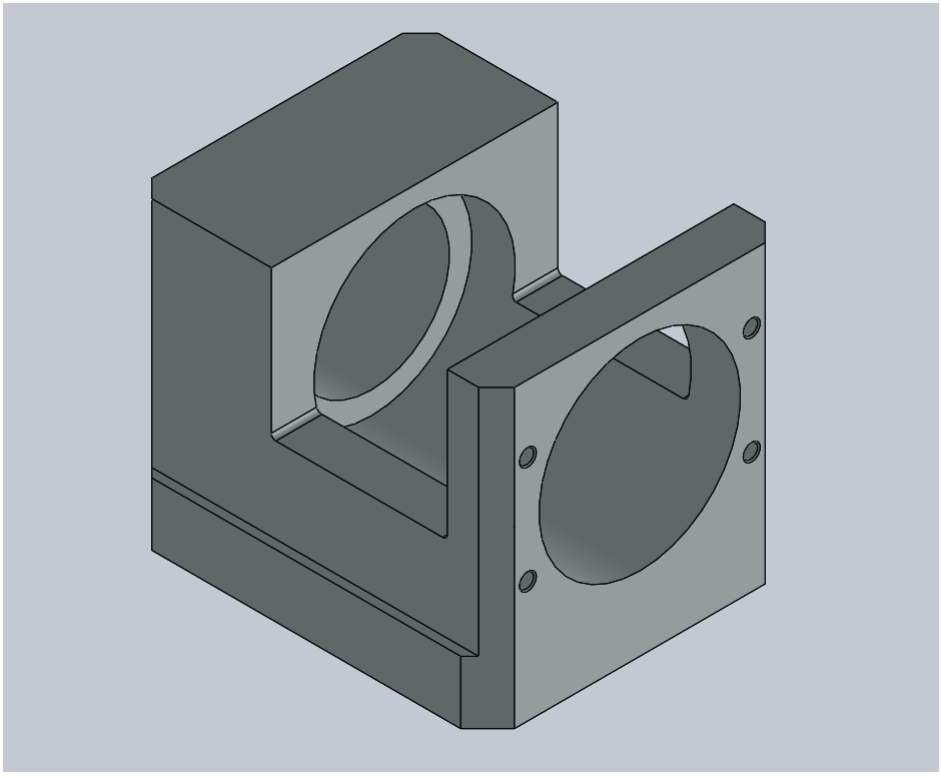


Fig. 8: *Gearbox mount*

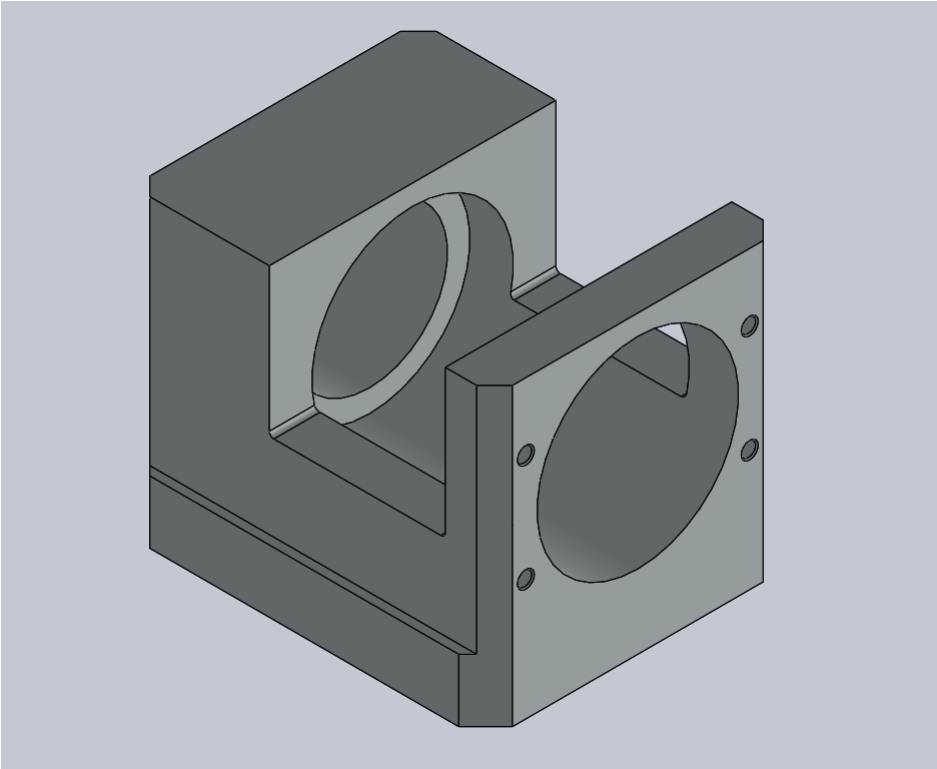


Fig. 9: *Gearbox mount*

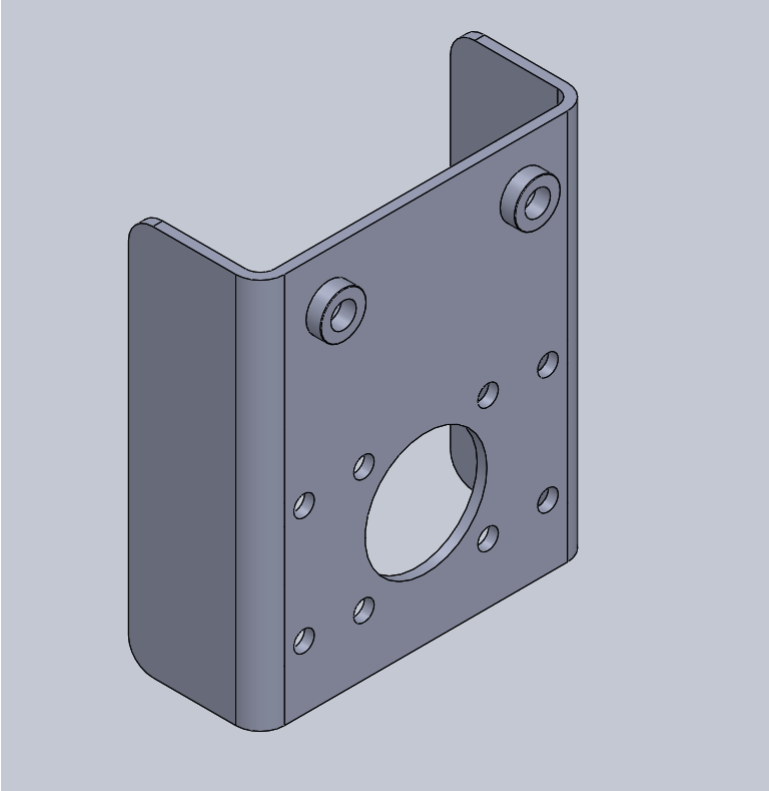


Fig. 10: *Mount plate 1*

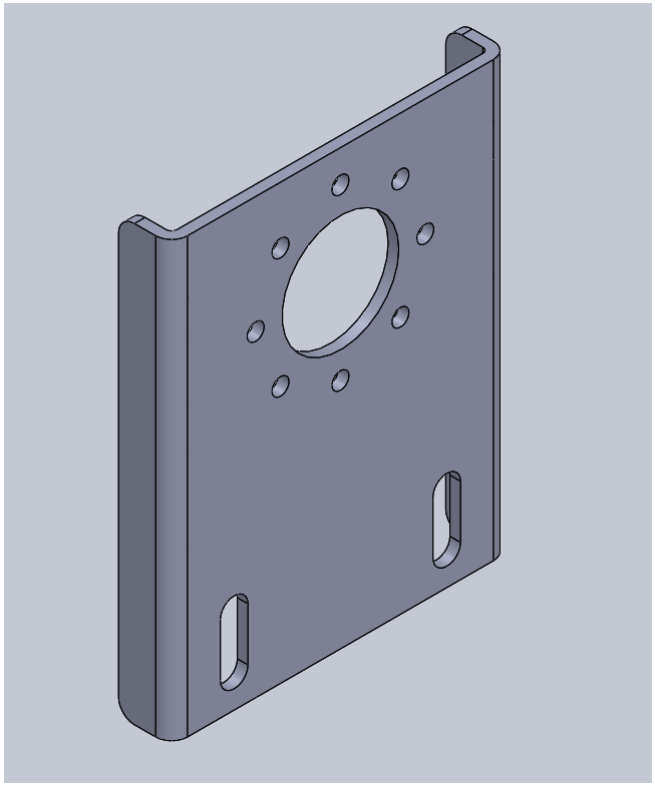


Fig. 11: *Mount plate 2 with adjustable belt tension*

2.4 Calculation and selection for motors

We use 125 (mm) Rotacaster Omni wheel with the specification of:

-Load of each wheel:

-Diameter of the wheel: 125 (mm)

Specification of the robot:

-Weight of the robot: $m = 22.8$ (kg)

-Diameter of the wheel: $2r = 125$ (mm)

-Velocity of the robot: $v = 0.5$ (m/s)

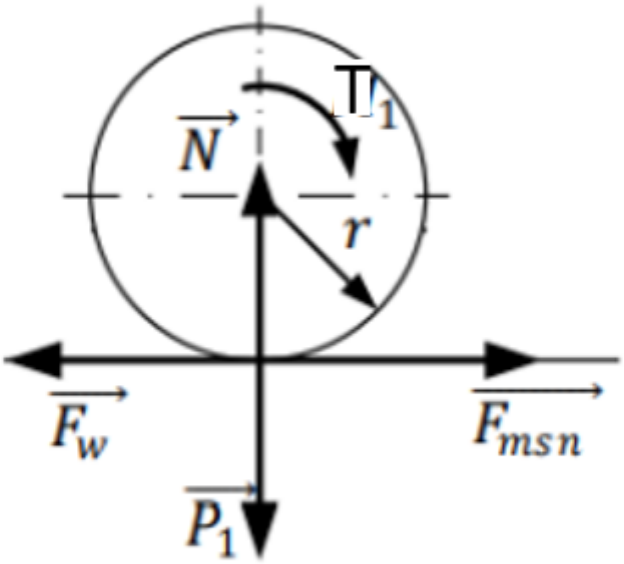


Fig. 12: *Analysis of the total forces acting on a wheel*

With:

-$\overrightarrow{F_{fric}}$: Static friction force between Omni wheel and dry concrete surface.

-$\overrightarrow{F_w}$: Pulling force generated by the motor torque.

-$\overrightarrow{P_1}$: Weight of a wheel, $P_1 = P/3$

-$\overrightarrow{N}$: Normal force exerted by the floor.

-T_1: Torque generated by the motor.

-m_1: Load-bearing mass, $m_1 = m/3$

The driving wheel moves forward at the same velocity $v = 0.5$ (m/s). The rotational speed of the wheel is:

$$n = \frac{60.1000.v}{2\pi r} = \frac{60.1000 \times 0.5}{125\pi} = 76.4 \text{ (rpm)}$$

The equation of force equilibrium is:

$\overrightarrow{F_{fric}} + \overrightarrow{F_w} + \overrightarrow{N} + \overrightarrow{P_1} = m_1 \overrightarrow{a}$

With $\overrightarrow{a}$ is the wheel acceleration

•When Robotino is in uniform motion

The velocity of the mobile robot remains constant at $v = 0.5$ m/s, therefore the acceleration a is constant and equal to $a = 0.$ (m/s^2). Then:

$$N = P_1 = m_1 g = \frac{22.8 \times 9.8}{3} = 74.48 \text{ (N)}$$

The pulling force generated by the motor torque is:

$$F_w = F_{fric} = \mu.N = 0.015.74.48 = 1.1172 \text{ (N)}$$

With:

$\mu = 0.015$, the coefficient of rolling friction when moving on a concrete floor surface.

Power on the wheel axle with no acceleration and velocity of $v = 0.5$ (m/s) is:

$$P_{wheel} = F_w.v = 1.1172 \times 0.5 = 0.5586 \text{ (W)}$$

•When Robotino accelerates:

The pulling force generated by the motor torque:

$$F_w = F_{fric} + m_1.a = \mu.N + m_1.a = 0.9 \times 74.48 + \frac{22.8}{3} \times 0.3 = 69.312$$

With:

$a = 0.3$ (m/s^2), the acceleration when the vehicle is accelerating.

$\mu = 0.9$: coefficient of rolling friction when moving on a concrete floor surface.

Wheel's shaft torque with acceleration:

$$T_1 = F_w.r = 69.312 \times 0.0625 = 4.332 \text{ (N.m)}$$

Power on the wheel axle to reach the velocity of $v = 0.5$ (m/s) is:

$$P_{wheel} = \frac{T_1 \times n}{9.55} = \frac{4.332 \times 76.4}{9.55} = 69.312 \times 0.5 = 34.656 \text{ (W)}$$

A belt drive module consists of a gearbox with ratio 1:8 and a belt drive with ratio of 1:4 => The total reduction ratio between the motor and the wheel is 1:32

$$n_{motor} = n \times 32 = 76.4 \times 32 = 2444.8 \text{ (rpm)}$$

Power on the motor shaft:

$$P_{motor} = \frac{P_{wheel}}{\eta} = \frac{34.656}{0.855} = 40.53 \text{ (W)}$$

$\eta = \eta_{belt} \times \eta_{gearbox} = 0.95 \times 0.9 = 0.855$

$\eta_{belt} = 0.95$ (0.95 ÷0.96) - belt transmission efficiency.

$\eta_{gearbox} = 0.9$ - gearbox efficiency.

Motor shaft's torque:

$$T_{motor} = \frac{9.55 \times P_{motor}}{n_{motor}} = \frac{9.55 \times 40.53}{2444.8} = 0.158 \text{ (N.m)}$$

No.	**Technical Information**	**Value**
1	Power	40.53 (W)
2	Torque	0.158 (N.m)
3	Speed	2444.8 (rpm)

Table 3: *Motor requirement specifications*

Choose the motor: GR 42x40

Data	GR 42x40
Rated voltage (VDC)	24
Starting torque (N.m)	0.33
Continuos rated speed (rpm)	3100
Continuous current (A)	1.2
No load speed (rpm)	3800
No load current (A)	0.18

Table 4: *Motor GR 42x40 specifications*

2.5 Base design

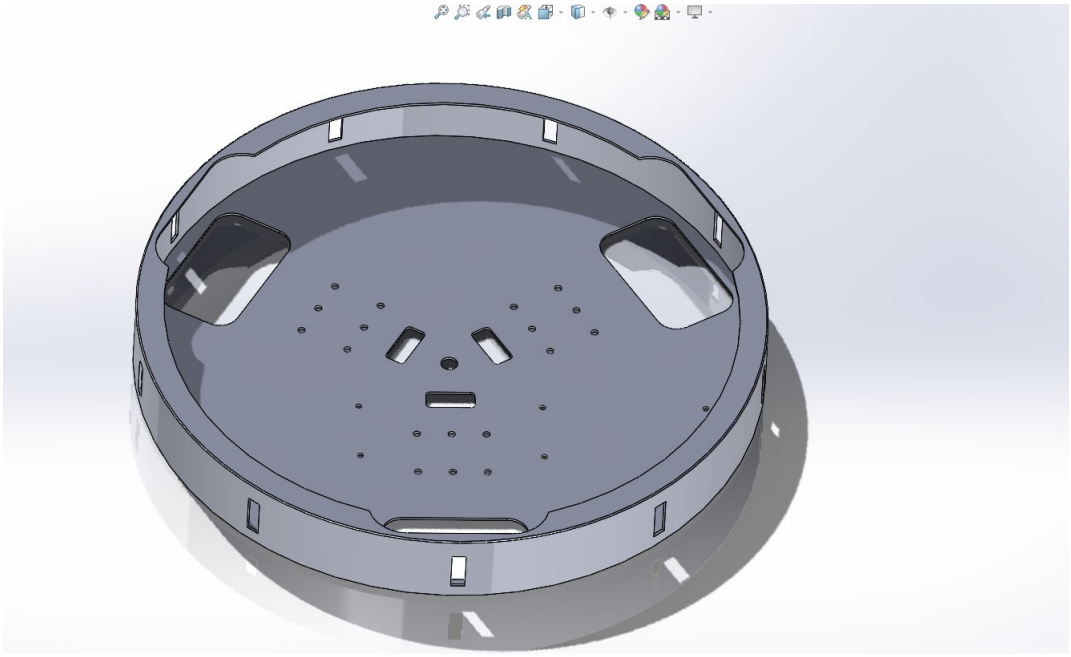


Fig. 13: *Main bearing base frame*

The entire base frame is designed entirely using a circular shape with diameter of 450mm of stainless steel, this type of steel has the material stainless steel PA6, tensile stress greater than 80 MPA. For ease of calculation, we consider the frame as a homogeneous block. Consider the entire robot load of 30kg. In this case of simulating bending strength using SolidWorks software, consider a load of 30kg evenly distributed over the entire frame. Since only the frame is included, the 3 wheels will be removed and replaced with 3 stainless steel plate that mount on the exact place where the wheels are, which simulate the load-mounting part instead of the wheels.

Simulation result

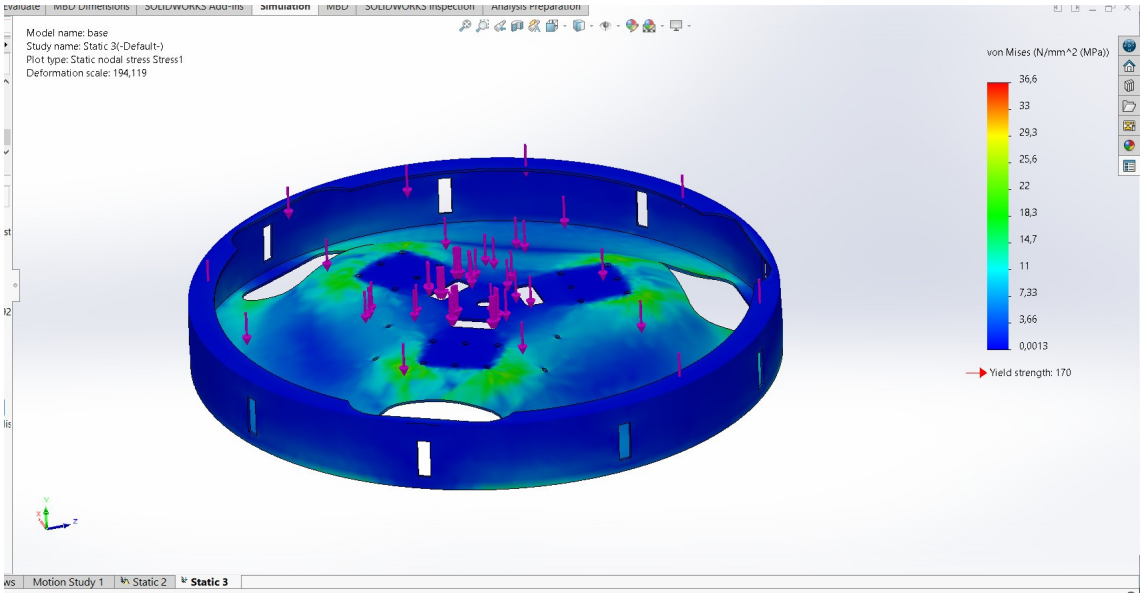


Fig. 14: *Yield stregth (MPA)*

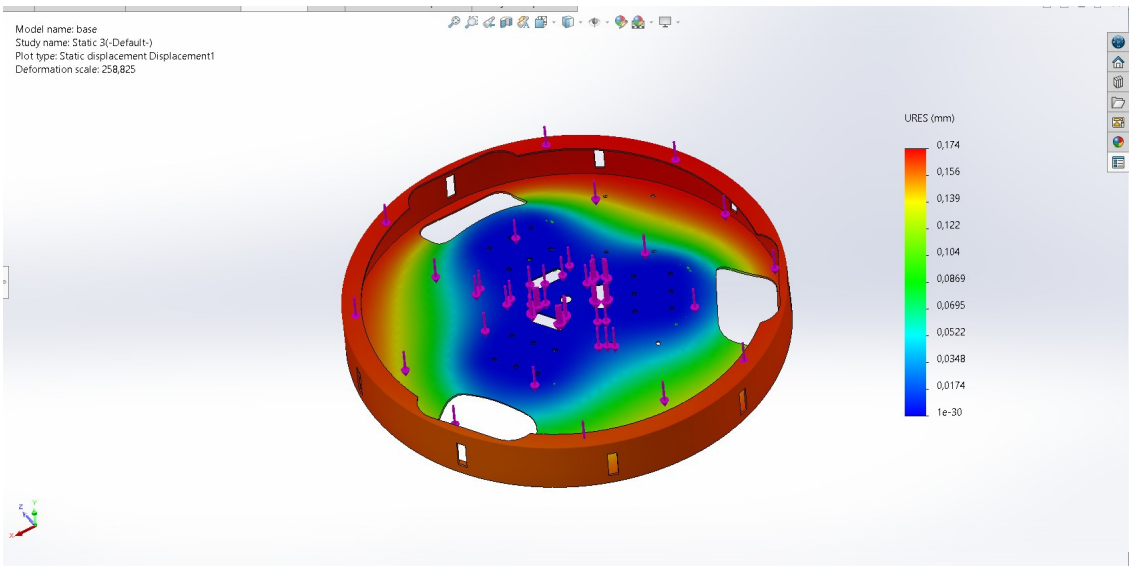


Fig. 15: *Base displacement (mm)*

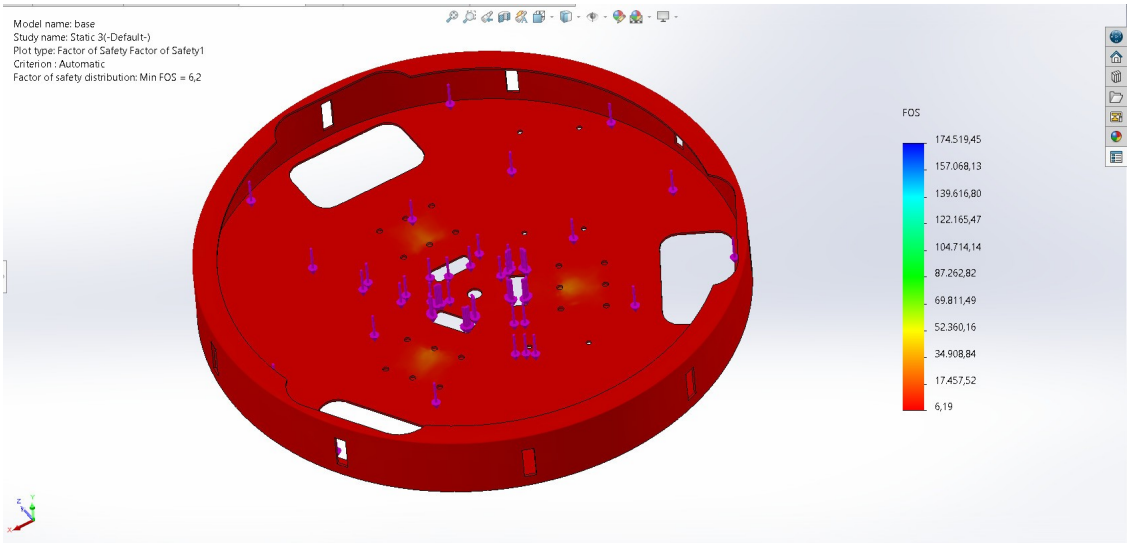


Fig. 16: *Factor of safety*

From the simulation, the displacement of the base frame is approximately 0.17 mm, which is not significant.

When approach the factor of safety, we have the following formula:

$$FOS = \frac{YieldStress}{WorkingorDesignstress}$$

For use with ordinary materials where loading and environmental conditions are not severe, the FOS range between 2-2.5. Therefore we choose the FOS for our base frame equal 2.5. The Factor of safety plot in Solidwork in a condition of 30kg load result in the Min FOS of 6.2, which is significantly higher than our 2.5.

=> Base on those result, we can conclude that our base frame is durable.

3 ELECTRICAL AND ELECTRONIC SYSTEM

3.1 Signal transmission system of Robotino

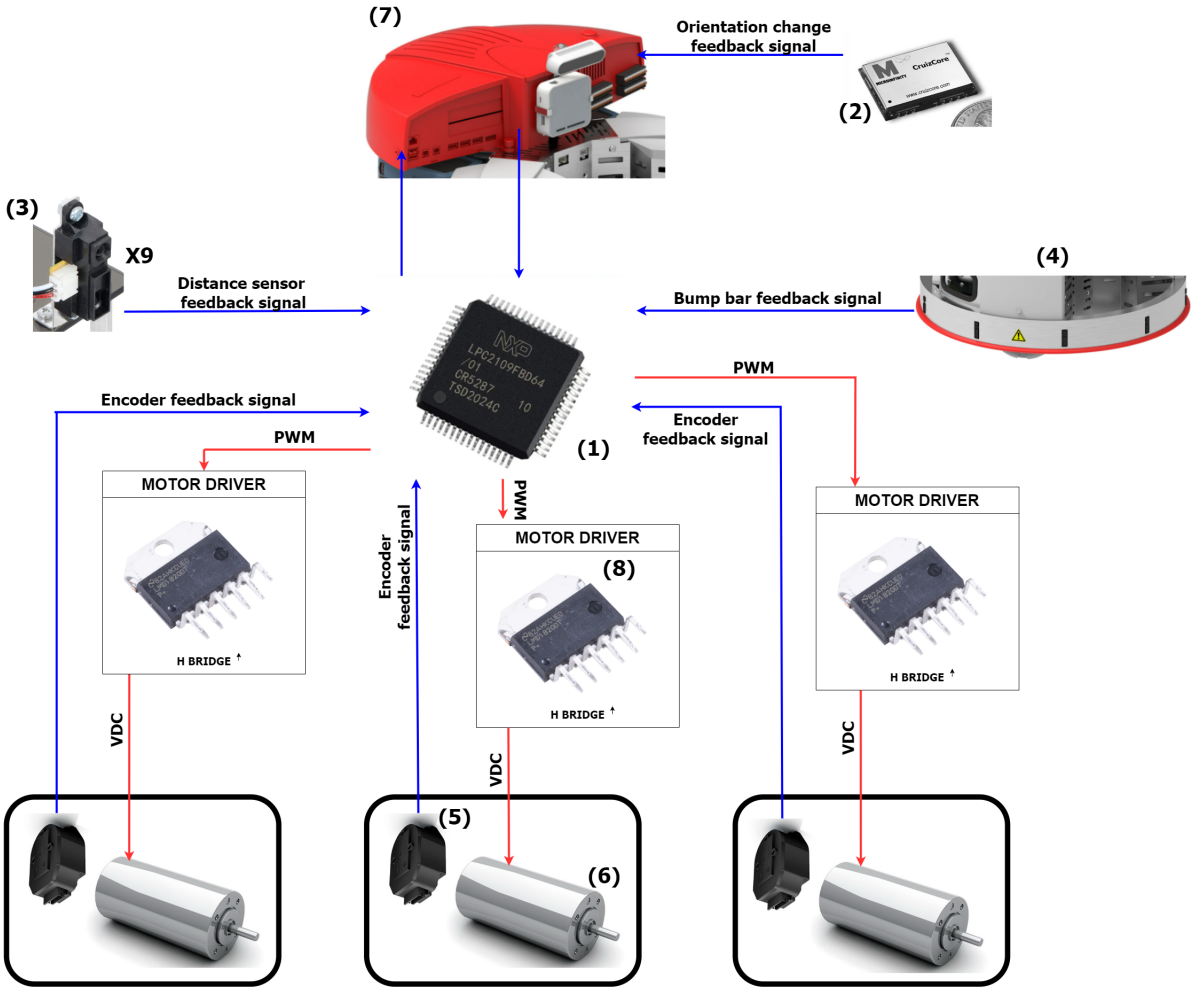


Fig. 17: *Signal diagram*

Note

(1): ARM Microcontroller for processing feedback signal from sensors, encoder, output PWM for motor control and communicate with Control Unit.

(2): CruizCore R6093U gyroscope, determines the change in the orientation of Robotino.

(3): Distance sensor GP2Y0A41SK0F to detect object from distance.

(4): SK EKS011 bumper bar ensures that in the event of a collision, program execution and movement are interrupted.

(5): The incremental encoders RE 30-2-500, based on the feedback signal, we can adjust the real motor speed to our desired and for determine the position of mobile robot.

(6): DC motors of type Dunkermotoren GR 42x40.

(7): Control Unit, Embedded PC to COM Express specifications, CPU Intel i5 8th generation

(8): Motor driver, H-bridge LMD18200 IC enables bidirectional control of DC motors.

3.2 Electrical system

Electrical general diagram

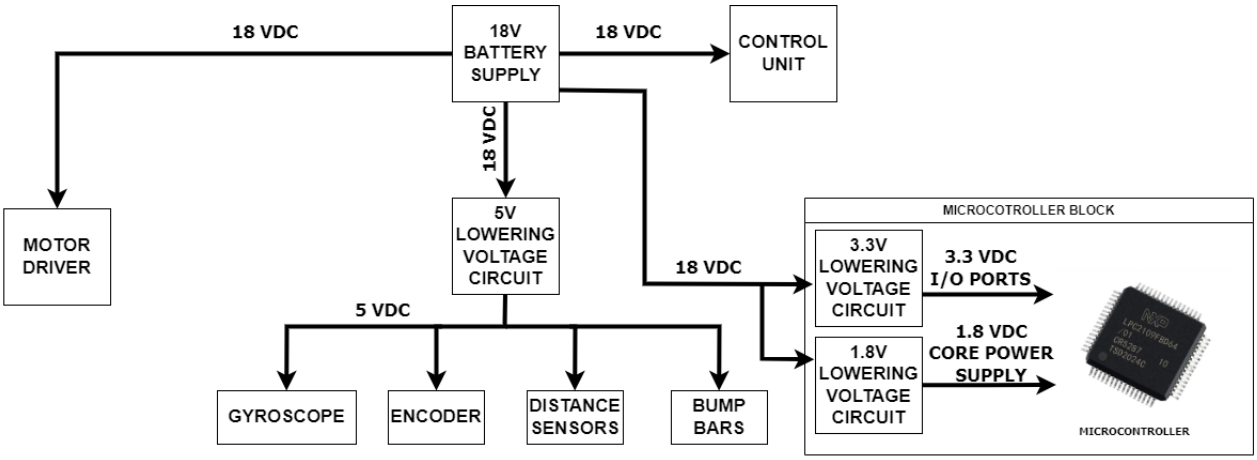


Fig. 18: *Electrical circuit diagram*

The Robot motor uses 24VDC power, through a 1.8V low-voltage circuit to supply the NXP LPC2109FBD64 microcontroller, 3.3V lower voltage circuit for I/O ports, and lastly lowering to 5V for the integrated peripherals, such as Gyroscope, encoders, ...

3.3 Electronic system

Control unit - Embedded PC

Fig. 19: *Control unit of Robotino*
[5]

When dealing with intricate demands, the microcontroller alone may struggle to fulfill the desired operations effectively. In addition to intricate computations, multitasking is also necessary, necessitating the use of computers to optimize the robot's performance. The robotino is equipped with an embedded PC, running on Linu Ubuntu operating system, which sastified the demand when operating the robot.

Parameter	Value
Type	Embedded PC to COM Express specifications
Operating system	Linux Ubuntu 18.04 LTS (64 bit)
CPU	Intel i5 8th generation, 2.5 GHz clock, up to 4.2 GHz in Turbo mode, 4 physical cores with hyperthreading
Random access memory	8 GB RAM
Hard disc	64 GB SSD

Table 5: *Embedded PC specifications*

Microcontroller - NXP LPC2109FBD64

Fig. 20: *Microcontroller of Robotino*
[6]

Integrating an additional microcontroller with the embedded PC in a mobile robot enables real-time motor control, relieving the PC from low-level tasks and reducing processing load. The microcontroller offers dedicated motor control capabilities, ensuring precise and responsive control over motor functions. It enhances fault tolerance by allowing the microcontroller to independently monitor and control motors if the PC encounters issues. Additionally, the use of a separate microcontroller facilitates modularity and expandability, enabling easy integration of additional motors or peripheral devices in the future. This combination optimizes motor control, overall performance, and flexibility in mobile robot applications. For these reasons, the Robotino comes with a microcontroller LPC2109FBD64.

Product Attribute	Attribute Value
Manufacturer	NXP
Core	ARM7TDMI-S
Program Memory Size	64 kB
Maximum Clock Frequency	25 MHz
Number of I/Os	46 I/O
Data RAM Size	8 kB
Supply Voltage	1.65V - 1.95V

I/O Voltage	3.3V
Interface Type	CAN, SPI, UART

Table 6: *Microcontroller specifications*

CruizCore R6093U gyroscope

Robotino is equipped with a gyroscope, which is used to increase the accuracy of positioning. For this, the gyroscope determines the change in the orientation of Robotino. The gyroscope is installed and connected in the control unit on the control board. As soon as Robotino's operating system recognizes the gyroscope, its signal is used to correct the position determination based on the drive system (odometry). Programming by the user is not required. The technology of gyroscope is measuring angular velocities based on angular momentum conservation. If one attempts to rotate a rotating object about an axis other than that about which it rotates, it exerts a torque opposite to the motion, so that the total angular momentum is maintained. The forces generated by this torque are measured in a gyroscope at the suspension points of a gyroscope. Since mechanical gyroscopes are too large, heavy and fragile, digital gyroscopes have been developed in recent years that are no larger than a one-euro-coin but with extreme accuracy.

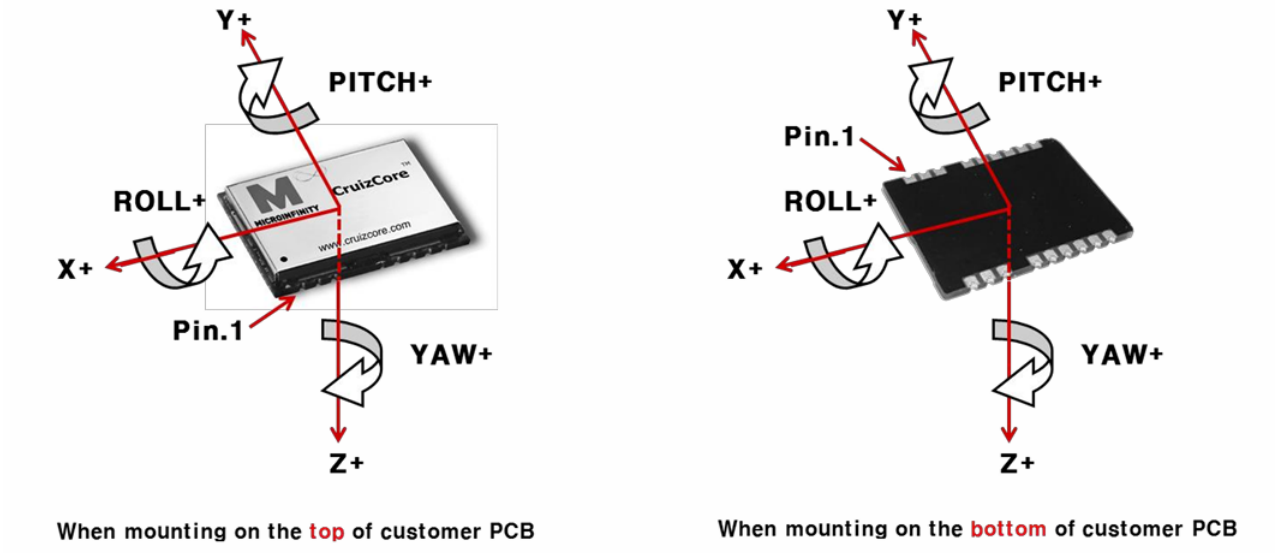


Fig. 21: *The CruizCore R6093U coordinate system*
[7]

Product Attribute	**Attribute Value**
Start-up time	0.5 sec
Angular rate	± 250 deg/sec
Acceleration	± 2 g
Angle resolution	0.01 deg/sec

Acceleration resolution	1 mg
Angular rate resolution	0.01 deg/sec
Supply Voltage	2.9V - 5.5V
Current	13 mA
Power Consumption	39 mW

Table 7: *Gyroscope specifications*

Distance sensor GP2Y0A41SK0F

The infrared distance sensors make it possible to determine the distance to objects around Robotino. Robotino has a total of nine infrared sensors arranged at an angle of 40 ° to each other around the base. Each distance sensor supplies a voltage value in volts whose magnitude depends on the distance to a reflecting object. They allows accurate relative distance measurements of an object between 4 cm and 30 cm. Its peculiarity is its easy connection, which consists only of power supply and an analogue output signal. Its evaluation electronics measures the distance and outputs it via an analogue voltage signal.

Product Attribute	**Attribute Value**
Operating voltage	4.5 V - 5.5 V
Average current consumption	12 mA
Distance measuring range	4 cm to 30 cm
Output type	Analog voltage
Output voltage differential over distance range	2.3 V
Update period	16.5 ± 4 ms
Size	44.5 mm × 18.9 mm × 13.5 mm
Weight	3.5 g

Table 8: *Distance sensor specifications*

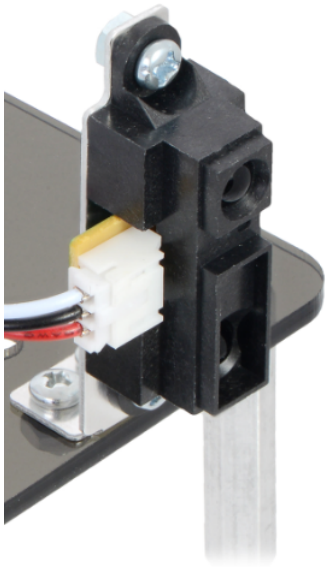


Fig. 22: *Distance sensor Sharp GP2Y0A41SK0F*
[8]

SK EKS011 bumper bar

Fig. 23: *SK EKS011 bumper bar*
[9]

The Robotino bumper bar ensures that in the event of a collision, program execution and movement are interrupted. The collision protection sensor of the Robotino is a so-called safety edge. This safety edge consists of a plastic profile of different shapes with integrated switching chamber. There are two separate conductive areas in the switching chamber. These are short-circuited when the safety edge is pressed and thus generate a signal for the evaluation device. The safety edge used in the Robotino operates according to the quiescent current principle so that a possible cable break can be detected and the Robotino can be stopped. The quiescent current is an electrical current that flows continuously in a circuit, even if it is not active. If this current does not flow, there is a cable break or destruction of the safety edge.

Incremental encoder RE 30-2-500

Each Robotino motor is equipped with an incremental encoder. Based on the values of the incremental encoder, the motor controller can adjust and regulate the real motor speed at the desired speed. In addition, the incremental encoder values are used to determine the position of the mobile system.

Fig. 24: *Incremental encoder RE 30-2-500*
[10]

Product Attribute	**Attribute Value**
Operating voltage	5V
Impulses per revolution	512 ppr
Signal rise time	180ns
Current consumption	max 0.4 (3.9 mA)

Table 9: *Incremental encoder RE 30-2-500 specifications*

Motor driver - H bridge LMD18200

The Robotino utilizes the H-bridge integrated circuit (IC) LMD18200 as its motor driver. The LMD18200 is a versatile and efficient H-bridge driver that enables bidirectional control of DC motors. With its robust design and high current capability, it is well-suited for driving the motors in the Robotino. The LMD18200 IC provides features like thermal shutdown protection, internal clamp diodes, and adjustable current limiting, ensuring safe and reliable motor operation. Its compact size and ease of integration make it an ideal choice for the space-constrained Robotino platform. By leveraging the capabilities of the LMD18200, the Robotino can achieve precise motor control, efficient movement, and reliable performance in a wide range of applications.

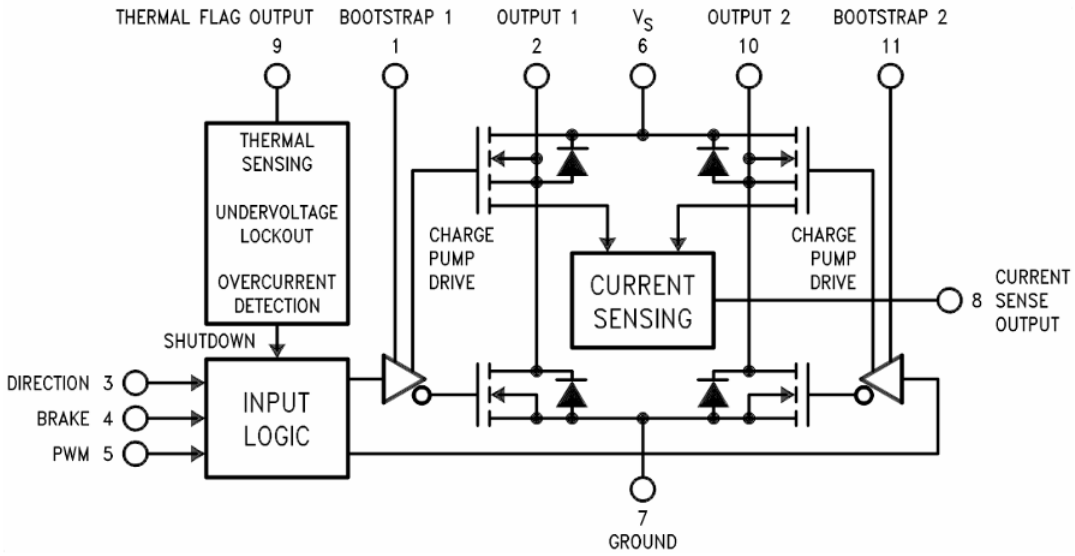


Fig. 25: *Functional Block Diagram of LMD18200* [11]

Absolute Maximum Ratings	**Value**
Total Supply Voltage (VS, Pin6)	12V - 60V
Voltage at Pins 3,4,5,8 and 9	12V
Signal rise time	180ns
Peak Output Current (200ms)	6A
Continuous Output Current	3A
Power Dissipation	25W

Table 10: *H-bridge LMD18200 specifications*

Power supply - Festool 18V 5.2 Ah Li-ion battery

Fig. 26: *Festool 18V 5.2 Ah Li-ion battery*
[12]

Up to 4 lithium-ion high-performance rechargeable batteries from Festool connected in parallel supply the Robotino with 18 V DC with a battery life of up to 10 hours. When empty, the empty batteries can be charged with the Festool TCL 6 or SCA 8 quick chargers.

Product Attributes	**Attributes Value**
Voltage (rated voltage)	18V
Capacity	5.00 Ah
Product weight without accessories	0.70 kg

Table 11: *Festool 18V 5.2 Ah Li-ion battery specifications*

4 MODEL AND CONTROL ALGORITHM DESIGN OF THE SYSTEM

4.1 Kinematic equation analysis :

- Analyzing kinematic of a general vehicle system.:

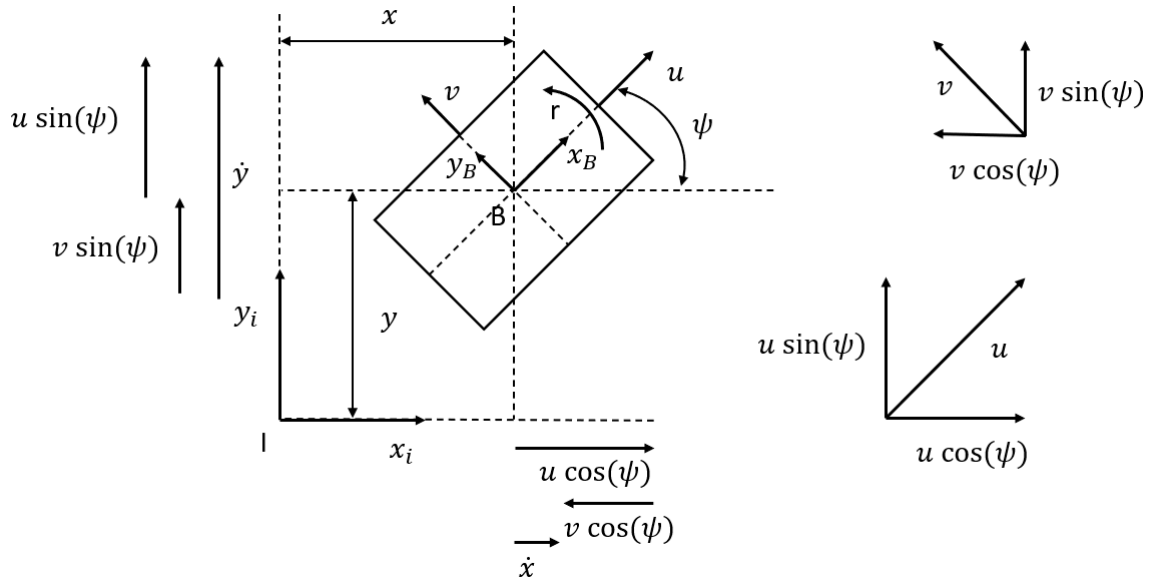


Fig. 27: Analyzing kinematic on a general Model

x : Forward displacement of the mobile robot w.r.t
y : Lateral displacement of the mobile robot w.r.t
ψ : Angular displacement of mobile robot w.r.t
u : Forward velocity of the mobile robot w.r.t
v : Lateral velocity of the mobile robot w.r.t
u : Angular velocity of the mobile robot w.r.t

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} u \times cos(\psi) - v \times sin(\psi) \\ u \times sin(\psi) - v \times cos(\psi) \\ r \end{bmatrix}$$

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} u \times cos(\psi) - v \times sin(\psi) \\ u \times sin(\psi) - v \times cos(\psi) \\ r \end{bmatrix} = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} u \\ v \\ r \end{bmatrix}$$

$$(\dot{\eta}) = J(\psi) \times \zeta$$

- Forward differential kinematics :

+ For given velocity commands , finding the derivatives of generalized coordinates(finding the system's motion)

$$\dot{\eta} = J(\psi) \times \zeta$$

-Inverse differential kinematics :

+ For the desired(given) derivatives of generalized coordinates (or given position trajectory), finding the corresponding velocity input commands.

$$\zeta = J^{-1}(\psi) \times \dot{\eta}$$

- A Generalized Wheel Model (Kinematics):

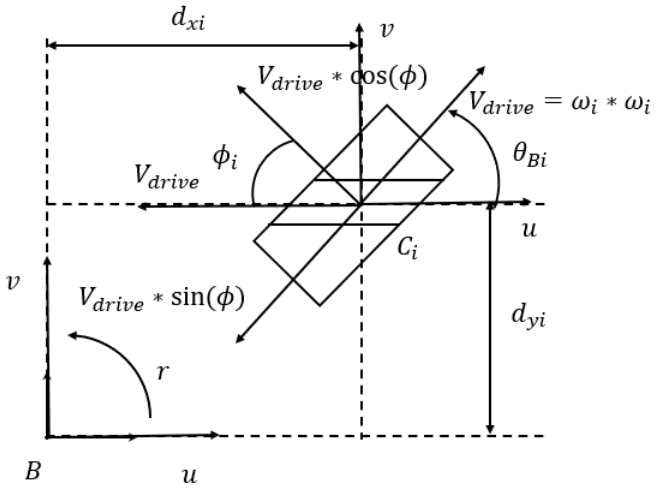


Fig. 28: *Analyzing kinematic on a general wheel*

$$\omega_i = \begin{bmatrix} \frac{1}{a_i} & \frac{1}{a_i} \times tan(\phi_i) \end{bmatrix} \times \begin{bmatrix} \dot{x_{ci}} \\ \dot{y_{ci}} \end{bmatrix}$$

$$u - rd_{yi} = \dot{x_{ci}} cos\theta_{Bi} - \dot{y_{ci}} sin\theta_{Bi}$$
$$v + rd_{xi} = \dot{x_{ci}} sin\theta_{Bi} - \dot{y_{ci}} cos\theta_{Bi}$$

$$\longrightarrow \begin{bmatrix} 1 & 0 & -d_{yi} \\ 0 & 1 & d_{xi} \end{bmatrix} \times \begin{bmatrix} u \\ v \\ r \end{bmatrix} = \begin{bmatrix} cos\theta_{Bi} & sin\theta_{Bi} \\ -sin\theta_{Bi} & cos\theta_{Bi} \end{bmatrix} \times \begin{bmatrix} \dot{x_{ci}} \\ \dot{y_{ci}} \end{bmatrix}$$

$$\omega_i = \begin{bmatrix} \frac{1}{a_i} & \frac{1}{a_i} \times tan(\phi_i) \end{bmatrix} \times \begin{bmatrix} cos\theta_{Bi} & sin\theta_{Bi} \\ -sin\theta_{Bi} & cos\theta_{Bi} \end{bmatrix} \times \begin{bmatrix} 1 & 0 & -d_{yi} \\ 0 & 1 & d_{xi} \end{bmatrix} \times \begin{bmatrix} u \\ v \\ r \end{bmatrix}$$

ω_i : Angular velocity of the i^{th} wheel
a_i : Radius of i^{th} wheel.
θ_{Bi} : Angle between the vehicle frame (B) to the wheel frame (c_i)
d_{xi} : and d_{yi} are the position coordinates of c_i with reference to B.
ϕ_i : Angle between roller axis to the x_{ci} axis.
u : Forward velocity of the mobile robot w.r.t frame B.
v : Lateral velocity of the mobile robot w.r.t frame B.
r : Angular velocity of the mobile robot w.r.t frame B.

- Analyzing the kinematic of robotino :

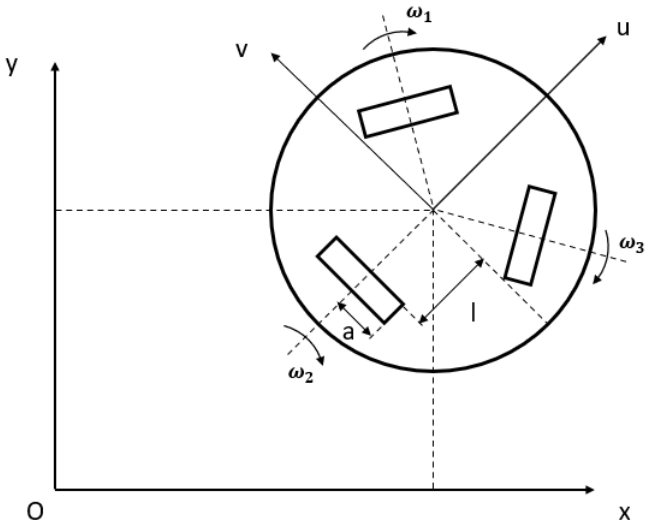


Fig. 29: *Analyzing kinematic on a robotino Model*

+ From the above robot model we have the following parameters:

$a_1 = a_2 = a_3 = a = 0.08$

$\theta_{B1} = 150°$,$\theta_{B2} = 270°$,$\theta_{B3} = 390°$

$l = 0.135$

$d_{x1} = l\cos(60°)$, $d_{x2} = l\cos(180°)$, $d_{x3} = l\cos(300°)$

$d_{y1} = l\sin(60°)$, $d_{y2} = l\sin(180°)$, $d_{y3} = l\sin(300°)$

$\phi_1 = \phi_2 = \phi_3 = 0°$

+ From the parameters, we can analyze the wheel kinematic equations as follows:

$$\omega_1 = \begin{bmatrix} \frac{1}{a} & 0 \end{bmatrix} \times \begin{bmatrix} -\frac{\sqrt{3}}{2} & \frac{1}{2} \\ -\frac{1}{2} & -\frac{\sqrt{3}}{2} \end{bmatrix} \times \begin{bmatrix} 1 & 0 & \frac{-\sqrt{3}l}{2} \\ 0 & 1 & \frac{l}{2} \end{bmatrix} \times \begin{bmatrix} u \\ v \\ r \end{bmatrix} = \frac{1}{2a}(-\sqrt{3}u + v + 2lr) \quad (1)$$

$$\omega_2 = \begin{bmatrix} \frac{1}{a} & 0 \end{bmatrix} \times \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & -l \end{bmatrix} \times \begin{bmatrix} u \\ v \\ r \end{bmatrix} = \frac{1}{a}(lr - v) \quad (2)$$

$$\omega_3 = \begin{bmatrix} \frac{1}{a} & 0 \end{bmatrix} \times \begin{bmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} \end{bmatrix} \times \begin{bmatrix} 1 & 0 & \frac{\sqrt{3}l}{2} \\ 0 & 1 & \frac{l}{2} \end{bmatrix} \times \begin{bmatrix} u \\ v \\ r \end{bmatrix} = \frac{1}{2a} \times (\sqrt{3}u + v + 2lr) \quad (3)$$

+ From (1), (2) and (3) we have a matrix showing the relationship between tricycle angular velocity and longitudinal velocity of vehicle:

$$\begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} = \begin{bmatrix}\frac{-\sqrt{3}}{2a} & \frac{1}{2a} & \frac{l}{a}\\ 0 & \frac{-1}{2a} & \frac{l}{a}\\ \frac{\sqrt{3}}{2a} & \frac{1}{2a} & \frac{l}{a}\end{bmatrix} \times \begin{bmatrix}u\\ v\\ r\end{bmatrix}$$

$$\longrightarrow \begin{bmatrix}u\\ v\\ r\end{bmatrix} = \zeta = \begin{bmatrix}-\frac{a\sqrt{3}}{3} & 0 & \frac{a\sqrt{3}}{3}\\ \frac{a}{2} & -a & -\frac{a}{2}\\ \frac{a}{4l} & \frac{a}{2l} & \frac{a}{4l}\end{bmatrix} \times \begin{bmatrix}\omega_1\\ \omega_2\\ \omega_3\end{bmatrix} = W\omega$$

4.2 Dynamic equation analysis :

- Introduction to Robot Dynamics :

+ Dynamics is the study of systems that undergo changes of state as time evolves. In mechanical systems such as robot, the change of states involves motion.

+ Is the science of motion . It describes why and how a motion occurs when forces and moments are applied on massive bodies. The motion can be considered as evolution of the position, orientation and their time derivatives.

+ Derivation of equations of motions for the system is the main step in dynamic analysis, since equations of motions are essential in design, analysis, and the control of the system.

- Equation of Motion (EoM)

+ The way in which the motion of the robotic system aries from the torques/forces applied by the actuators, or from external forces/moments applied to the system.

$$\tau = fun(\eta, \dot{\eta}, \ddot{\eta})$$

- Forward dynamics:

+ For a given input vector, τ,calculating the resulting motion of the robot, that is $\eta, \dot{\eta}, \ddot{\eta}$.

- Inverse dynamics:

+ For a given desired trajectory, $\eta, \dot{\eta}, \ddot{\eta}$, find the requred input vector, τ.

- Solve the dynamic system using the Lagrange Euler method:

+ We have the following general vehicle model:

+ Linear velocities of the body frame :

$$\dot{x_c} = u - y_{bc}r$$
$$\dot{y_c} = v + x_{bc}r$$

+ The kinetic and potential energies of the model are analyzed as follows :

$$KE = \frac{1}{2}m(\dot{x_c}^2 + \dot{y_c}^2) + \frac{1}{3}I_z r^2$$
$$PE = 0$$
$$L = KE - PE$$

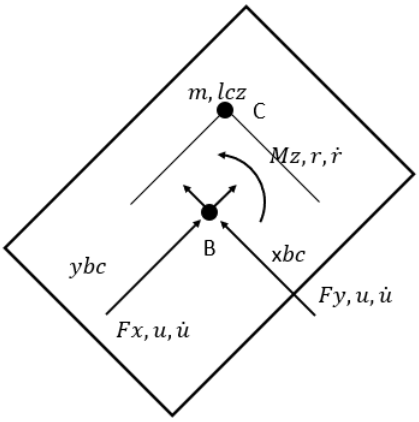


Fig. 30: *Analyzing Dynamic on a general Model*

$$\longrightarrow \tau_i = \frac{d}{dt}\frac{\partial L}{\partial \dot{q}i} - \frac{\partial L}{\partial qi}$$

+ The Lagrange Euler equation of the general vehicle model :

$$L = KE - PE = \frac{1}{2}m(\dot{x_c^2} + \dot{y_c^2}) + \frac{1}{2}I_z r^2$$

+ we have that :

$$Fx = \frac{d}{dt}\frac{\partial L}{\partial u}$$
$$Fy = \frac{d}{dt}\frac{\partial L}{\partial v}$$
$$Mz = \frac{d}{dt}\frac{\partial L}{\partial r}$$

+ Then we have the final equation of Fx, Fy and Mz :

$$Fx = m(\dot{u} - vr - x_{bc}r^2 - y_{bc}\dot{r})$$
$$Fy = m(\dot{v} - ur - y_{bc}r^2 + x_{bc}\dot{r})$$
$$Mz = I_{cz}\dot{r} + m(x_{bc}[\dot{v} + ur] - y_{bc}[\dot{u} - vr]) + m\dot{r}(x_{bc}^2 + y_{bc}^2)$$

+ Matrix of vector input τ:

$$\begin{bmatrix} m\dot{u} - mvr - mx_{bc}r^2 - my_{bc}\dot{r} \\ m\dot{v} + mur - my_{bc}r^2 + mx_{bc}\dot{r} \\ (I_c z + m(x_{bc}^2 + y_{bc}^2))\dot{r} + m(x_{bc}[\dot{v + ur}] - y_{bc}[\dot{u - vr}]) \end{bmatrix} = \begin{bmatrix} Fx \\ Fy \\ Mz \end{bmatrix}$$

$$\begin{bmatrix} m & 0 & -my_{bc} \\ 0 & m & mx_{bc} \\ -my_{bc} & mx_{bc} & I_{cz} + m(x_{bc}^2 + y_{bc}^2) \end{bmatrix} \times \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{r} \end{bmatrix} + \begin{bmatrix} -mr(v + x_{bc}r) \\ mr(u - y_{bc}r) \\ mr(x_{bc}u + y_{bc}v) \end{bmatrix} = \begin{bmatrix} Fx \\ Fy \\ Mz \end{bmatrix}$$

$$D\dot{\zeta} + n(\zeta) = \tau$$

-Mobile robot kinematic relation :

$$\dot{\eta} = \zeta(\eta)\zeta$$
$$\longrightarrow \zeta = J^{-1}(\eta)\dot{\eta}$$

+ Diffirentiating with respect to time, it gives :

$$\ddot{\eta} = J(\eta)\dot{\zeta} + \dot{J}(\eta)\zeta$$
$$\longrightarrow \dot{\zeta} = J^{-1}(\eta)[\ddot{\eta} - \dot{J}(\eta)\zeta]$$

+ Further, from the Principle of conservation of power/energy:

$$\tau = J^T(\eta)\tau_\eta$$

+ Mobile robot dynamic relation:

$$D\dot{\zeta} + n(\zeta) = \tau$$

- Dynamic analysis of Robotino :

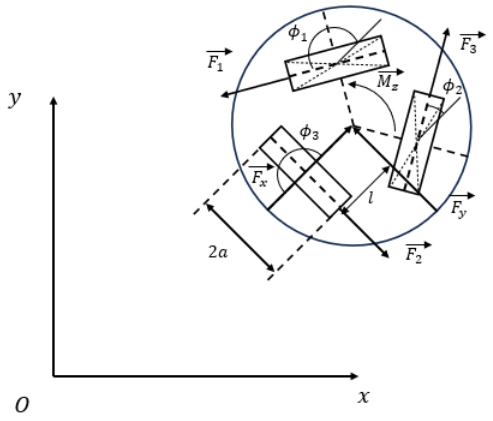


Fig. 31: *Analyzing Dynamic on robotino Model*

+ Forces analysis on model robotino :

$$Fx = F_1 cos(\phi_1) + F_2 cos(\phi_2) + F_3 cos(\phi_3)$$
$$Fy = F_1 sin(\phi_1) + F_2 sin(\phi_2) + F_3 sin(\phi_3)$$
$$Mz = (F_1 + F_2 + F_3)I$$

+ From there we have the following matrix:

$$\begin{bmatrix} Fx \\ Fy \\ Mz \end{bmatrix} = \begin{bmatrix} cos(\phi_1) & cos(\phi_2) & cos(\phi_3) \\ sin(\phi_1) & sin(\phi_2) & sin(\phi_3) \\ I & I & I \end{bmatrix} \times \begin{bmatrix} F_1 \\ F_2 \\ F_3 \end{bmatrix}$$

$$\tau = \Gamma\kappa$$

4.3 PID control algorithm:

- Introduction of the algorithm :

+ A proportional–integral–derivative controller (PID controller or three-term controller) is a control loop mechanism employing feedback that is widely used in industrial control systems and a variety of other applications requiring continuously modulated control. A PID controller continuously calculates an error value e(t) as the difference between a desired setpoint (SP) and a measured process variable (PV) and applies a correction based on proportional, integral, and derivative terms (denoted P, I, and D respectively)

- The mathematic of the PID controller :

$$u(t) = K_p e(t) + K_i \int_0^t e(t)d(t) + K_D \frac{de(t)}{dt}$$

+ When implementing the PID controller in practice, the input variable (error) is obtained by sampling the plant's output at the sample rate. Then, the PID algorithm is also calculated at the same rate. At the step k^{th}, we have:

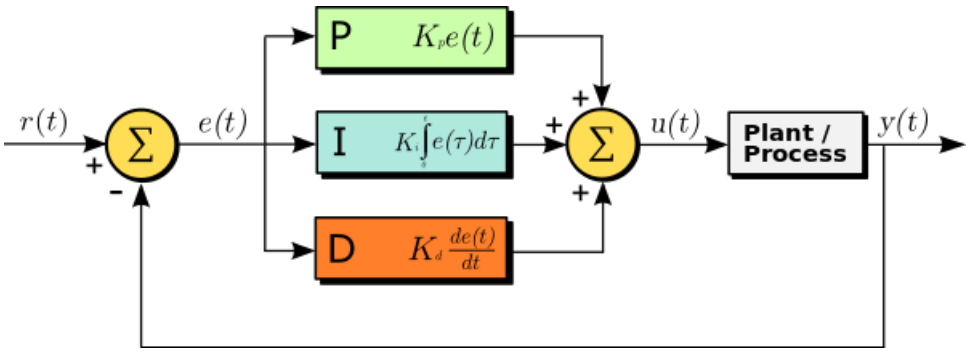


Fig. 32: *Block Diagram of PID Controller*

$$u_k = u_k^P + u_k^D + u_k^I$$

- P Calculation :

+ Term P is proportional to the current value of SP - PV error e(t).For example, if the error is large, the control output will be proportionately large by using the gain factor "K_P".Using the proportional control alone will result in an error between the set point and the process value because the controller requires an error to generate the proportional output response. In steady state process conditions an equilibrium is reached, with a steady SP-PV "offset".

$$u_k^P = K_P e_k$$

- D calculation :

+ Term D is a best estimate of the future trend of SP - PV error, based on its current rate of change. It is sometimes called "anticipatory control", as in effectively seeking to reduce the effect of the SP - PV by exerting a control influence generated by the rate of error change. The more rapid the change, the greater the controlling or damping effect.

$$u_k^D = K_D \frac{e_k - e_{k-1}}{T}$$

Low pass filter design for stage D:

+ Because T is a small number and looking at the calculation, the numerator of the equation is $(e_k - e_{k-1})$ .If a large disturbance occurs, D term will be very large, leading to fluctuations and damage to the system. Designing a low-pass filter for stage D will help significantly reduce the impact of noise and oscillation ,High noise and fluctuations in the system

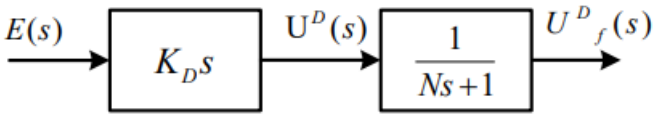


Fig. 33: *Low pass filter for D term*

+ mathemetic equation for Lowpass filter :

$$u_f^D(k) = \frac{N}{N+T}u_f^D(k-1) + \frac{T}{N+T}u^D(k)$$

Where $\alpha = \frac{T}{N+T}(0 < \alpha \leq 1)$: coefficient of low pass filter

+ We derive the equation of low-pass filter calculation :

$$\longrightarrow u_f^D(k) = (1-\alpha)u_f^D(k-1) + \alpha u_f^D(k)$$

- I calculation :

+ Term I accounts for past values of the SP - PV error and integrates them over time to produce the I term. For example if there is a residual SP - PV error after the application of proportional control, the integral term seeks to eliminate the residual error by adding a control effect due to the historic cumulative value of the error. When the error is eliminated, the integral term will cease to grow. This will result in the proportional effect diminishing as the error decreases, but this is compensated for by the growing integral effect.

+ We have three ways to calculate I term :

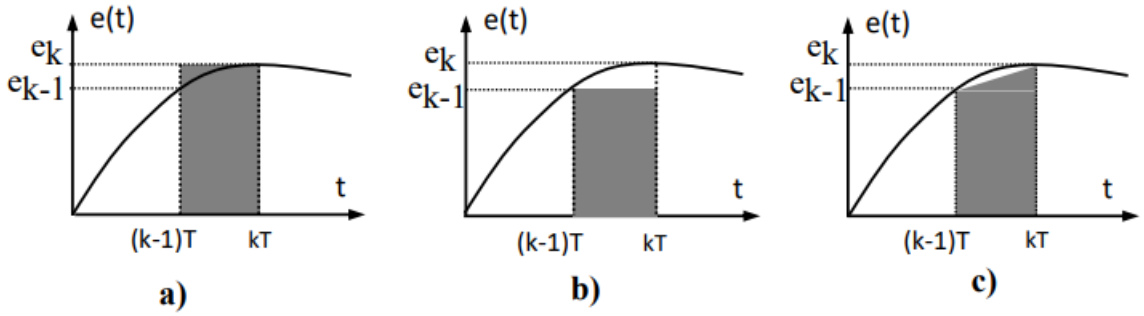


Fig. 34: *Integral approximation methods*

a)Backward rectangular approximation (backward Euler):

$u_k^I = K_I \Sigma_{i=1}^k T e_i = u_{k-1}^I + K_I T e_k$

b)Forward rectangular approximation(forward Euler):

$u_k^I = K_I \Sigma_{i=1}^k T e_{i-1} = u_{k-1}^I + K_I T e_{k-1}$

c)Trapezoidal approximation:

$u_k^I = K_I \Sigma_{i=1}^k T \frac{e_{i-1}+e_i}{2} = u_{k-1}^I + K_I T \frac{e_{k-1}+e_k}{2}$

+ Anti-windup design for term I :

+ From the block, we draw out the equation to calculate the anti-windup for I term

$$u^I(t) = \int_0^t [K_I e(t) + K_b e_{reset}(t)]dt$$
$$\longrightarrow u_k^I = u_{k-1}^I + K_I T e_k + K_b T e_k^{reset}$$

4.4 Position control (Point to point) :

Explaination of the symbols of the elements in the calculation of the Point-to-point position control algorithm :

$\eta_d(t)$ Desired positions

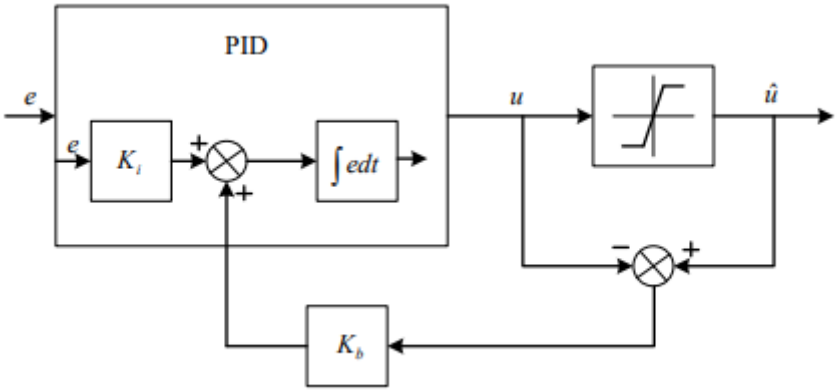


Fig. 35: *Block diagram of an anti-windup structure*

$\eta(t)$ Actual positions

$J(\eta), J^{-1}(\eta)$:Jacobian matrix

ζ :Control inputs

The vehicle control signal ζ will be calculated as follows :

$$\zeta = J^{-1}(\eta)[K\eta^{\sim}(t)]$$

- Design a controller for robotino :

+ First we have the error expressed by the following calculation with k sampling times:

$$\eta_k^{\sim} = \eta_k^d - \eta_k$$

+ After many tests, our team came to a conclusion when choosing PID controler for this point-to-point control problem with the following parameters:

+ About x axis x:

$K_{Px} = 1$

$K_{Ix} = 0$

$K_{Dx} = 0$

+ About y axis y:

$K_{Py} = 1$

$K_{Iy} = 0$

$K_{Dy} = 0$

+ About Angle of the robot psi:

$K_{P\psi} = 0.05$

$K_{I\psi} = 0$

$K_{D\psi} = 0$

+ Because all three controllers only use the proportional stage P, the control signal amplification matrix will be expressed as follows:

$$K = \begin{bmatrix} K_{Px} & 0 & 0 \\ 0 & K_{Py} & 0 \\ 0 & 0 & k_{P\psi} \end{bmatrix}$$

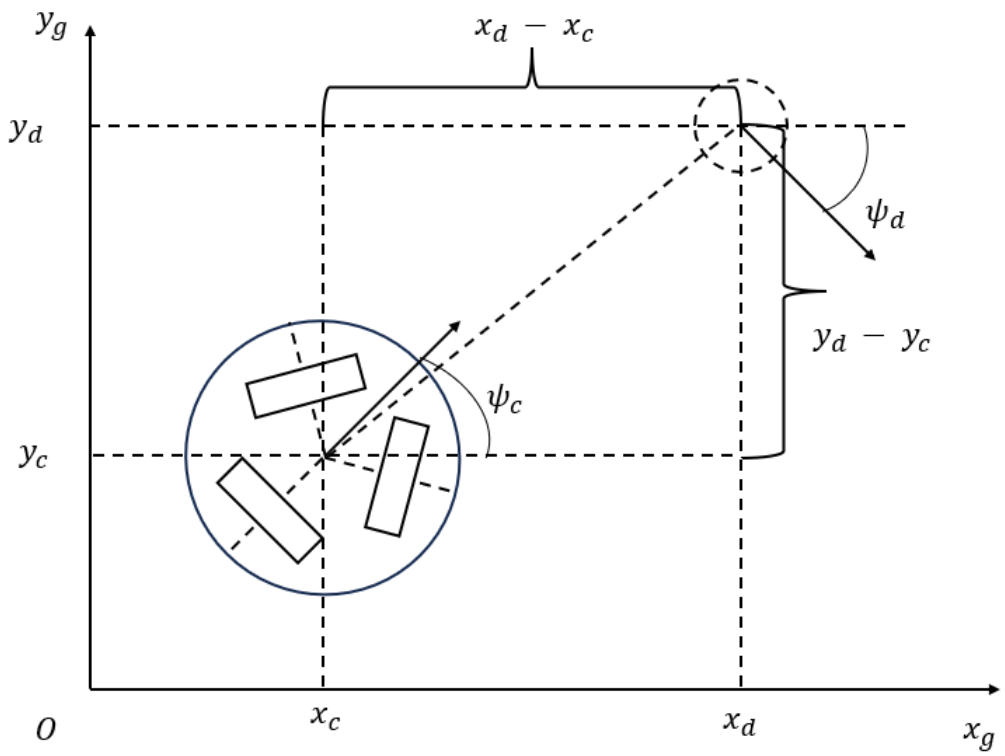


Fig. 36: *Point to point symbolic image*

+ From there we get the zeta control signal calculated as follows :

$$\zeta_k = J^{-1}(\psi_k)[K\eta_k^{\sim}] = \begin{bmatrix} cos(\psi) & sin(\psi) & 0 \\ -sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} K_{Px} & 0 & 0 \\ 0 & K_{Py} & 0 \\ 0 & 0 & k_{P\psi} \end{bmatrix} \times \begin{bmatrix} xd - x_k \\ yd - y_k \\ \psi d - \psi_k \end{bmatrix}$$

Based on this control input vector, we can find the individual wheel velocities of the mobile robot, as follows

$$\omega = W^{+}\zeta$$

From the analysis in the kinematics section, we have a matrix showing the relationship between the control signal ζ and the three-wheel angular velocity ω as:

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{a} & \frac{1}{a} \\ -\frac{\sqrt{3}}{2a} & \frac{1}{2a} & \frac{l}{a} \\ \frac{\sqrt{3}}{2a} & -\frac{1}{2a} & \frac{l}{a} \end{bmatrix} \times \begin{bmatrix} u \\ v \\ r \end{bmatrix}$$

Then, collect the encoder pulse after each sampling time T and convert it to the angular velocity unit (rad/s) through calculation. We get the actual velocity after each sampling time.

Applying it to the mathematical model, we get the actual velocity of the base:

$$\zeta = W\omega = \begin{bmatrix} u \\ v \\ r \end{bmatrix} = \begin{bmatrix} 0 & -\frac{a\sqrt{3}}{3} & \frac{a\sqrt{3}}{3} \\ \frac{2a}{3} & -\frac{a}{3} & -\frac{a}{3} \\ \frac{a}{3l} & \frac{a}{3l} & \frac{a}{3l} \end{bmatrix} \times \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}$$

Applying the Forward kinematic formula, we get the base's actual velocity compared to generalized coordinates:

$$\dot{\eta} = J(\psi) \times \zeta$$

Then apply the Forward Euler method to calculate the current position of the robot, the formula is expanded as follows where k is the number of sampling times every T seconds:

$$x_{k+1} = x_k + (u_k cos(\psi_k) - v_k sin(\psi_k))T$$
$$y_{k+1} = y_k + (u_k sin(\psi_k) + v_k cos(\psi_k))T$$
$$\psi_{k+1} = \psi_k + r_k T$$

- Block diagram for point-to-point algorithm in matlab :

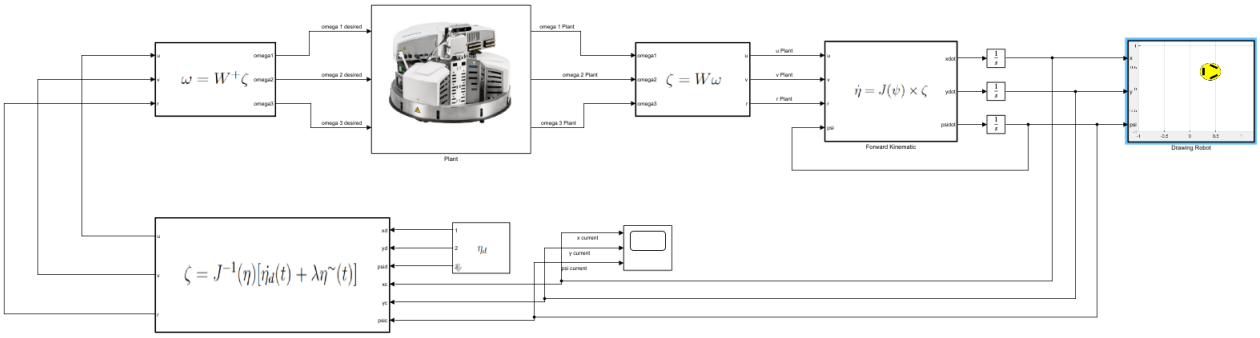


Fig. 37: *Block Diagram of Robotino's Point-to-Point Algorithm*

+ Set the setpoint values we give as $xd = 1, yd = 2$ and $psid = 1$ respectively :

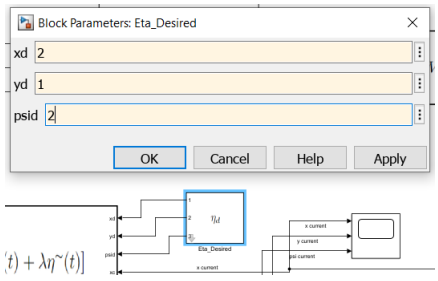


Fig. 38: *Setpoint in Robotino's Point-to-Point simulation*

- Robotino response chart in simulation :

+ Response to the position of the base with $\eta[2; 1; 2]$

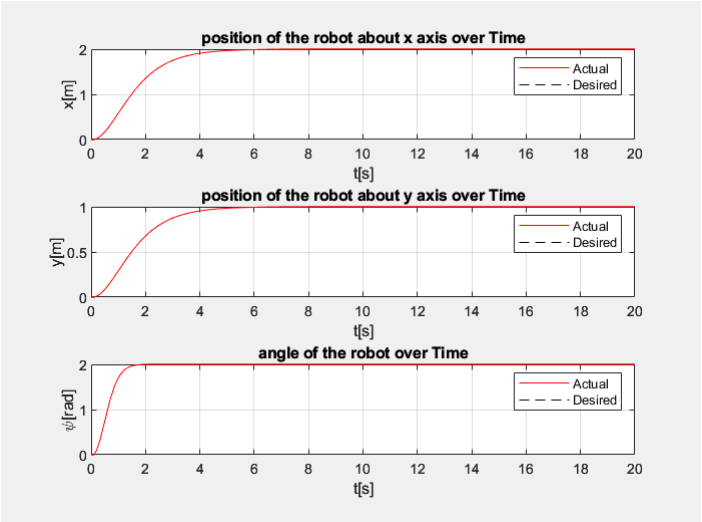


Fig. 39: *Position and pose of the robot over time*

+ The signal controls ζ over the base's timing during running :

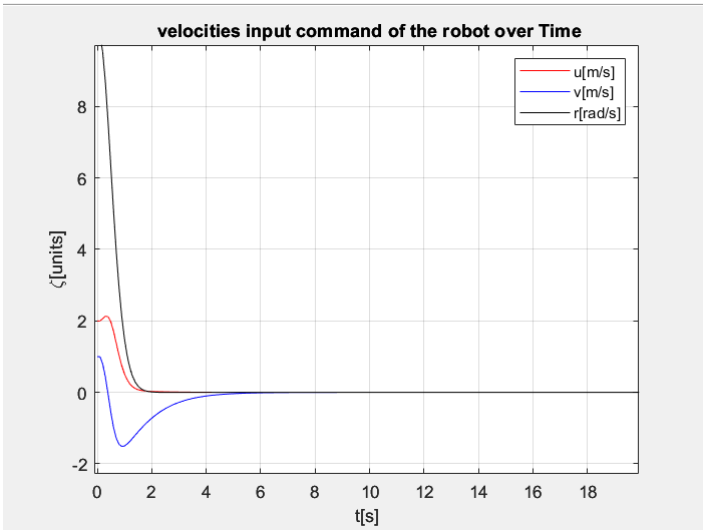


Fig. 40: *Control signal* ζ *of the robot over time*

6Angular velocity of each wheel ω over time of the sole during running :

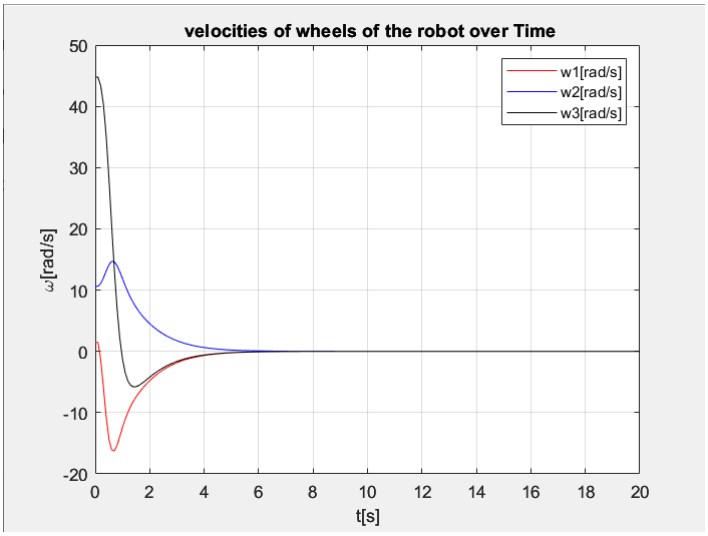


Fig. 41: *angular velocity* ω *of robot's wheels over time*

+ Simulate the point-to-point algorithm on matlab:

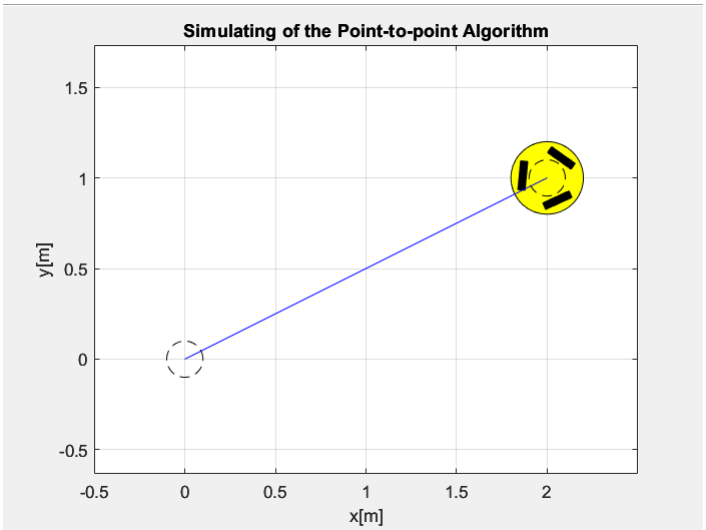


Fig. 42: *Simulating of the Point-to-point Algorithm in matlab*

4.5 Path Following algorithms :

- Introduction of path follow algorithms :

+ The path following algorithm typically takes into account the current position of the robot or vehicle, the desired path or trajectory, and sensor feedback to determine the appropriate control actions needed to stay on the path. It involves continuously adjusting the heading or orientation of the robot or vehicle and controlling its speed to ensure it stays aligned with the desired path.There are many algorithms such as :

+ Pure Pursuit Algorithm,Stanley Algorithm,Dynamic Window Approach,Model Predictive Control (MPC),...

- We chose the Pure Pursuit algorithm.The pure pursuit controller is an automatic steering method for wheeled mobile robots. It is a steering method, which means it computes the necessary angular velocity for a wheeled robot to stay on pre-computed paths. Linear velocity is assumed to be constant. Therefore, an additional velocity controller of your choice is needed if you wish to slow down the robot as it approaches the target

- Pure Pursuit algorithm :

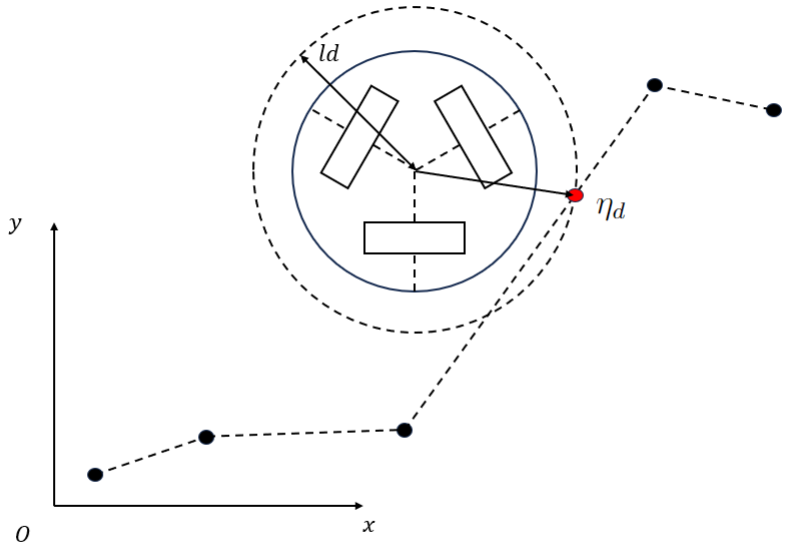


Fig. 43: *Visual Representation of the Pure Pursuit Algorithm*

ld : looking ahead distance

η_d Desired positions

+ The key of the pure pursuit controller involves calculating a goal point on the path that is a certain distance , which is called the look ahead distance, from the robot's current position.Then, the goal point is used to calculate the appropriate angular velocity needed for the robot to move toward that point.As the robot moves forward, a different goal point on the path is chosen and the robot's angular velocity gets updated. Since the robot can never reach this goal point and the goal point stays on the path, the end result would be the robot tracing the path.

+ We have an algorithm flow chart as follows :

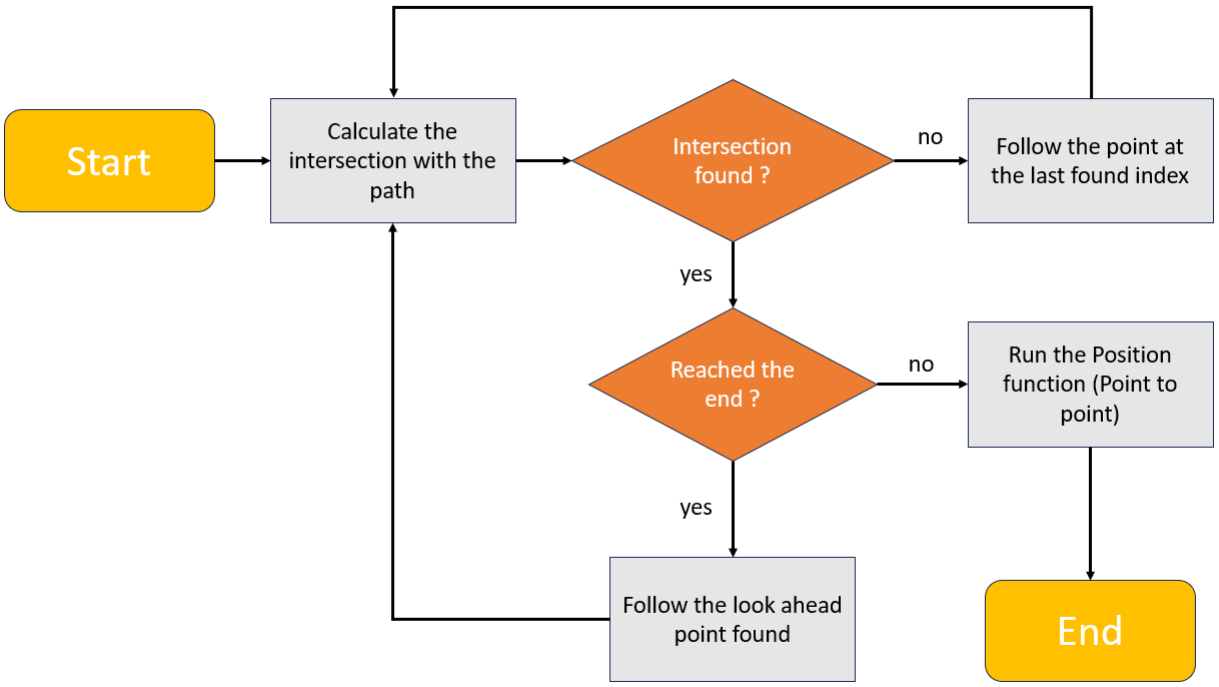


Fig. 44: Flowchart of the Pure Pursuit Algorithm

- Formulas for calculating intersection points of circles :

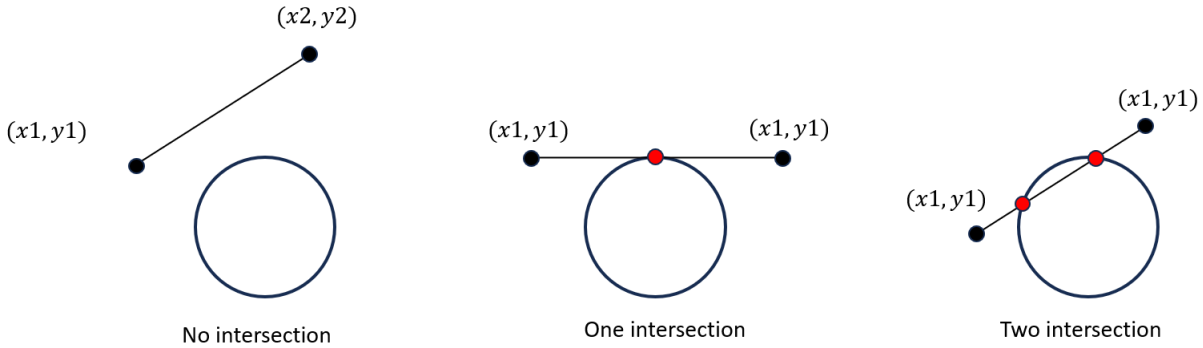


Fig. 45: Intersections of Lines and Circles

+ Technical declaration of :

$$d_x = x_2 - x_1$$
$$d_y = y_2 - y_1$$
$$d_r = \sqrt{d_x^2 + d_y^2}$$
$$D = x_1 y_2 - x_2 y_1$$

+ The points of intersection can be calculated as :

$$x = \frac{Dd_y \pm sgn^*(d_y) d_x \sqrt{r^2 dr^2 - D^2}}{d_r^2}$$
$$y = \frac{-Dd_x \pm |d_y| \sqrt{r^2 d_r^2 - D^2}}{d_r^2}$$

+ Where the function $sgn^*(x)$ is defined as :

$$sgn^*(x) = \begin{cases} -1 & \text{for } x < 0 \\ 1 & \text{otherwise} \end{cases}$$

+ The discriminant

$\Delta \equiv r^2 d_r^2 - D^2$

+ Therefore determines the incidence of the line and circle as summarized in the following table.

Δ	incidence
$\Delta < 0$	no intersection
$\Delta = 0$	tangent
$\Delta > 0$	intersection

- Matlab simulation and response of the Pure pursuilt algorithm :

+ The vehicle's trajectory is compared to the vehicle's set trajectory :

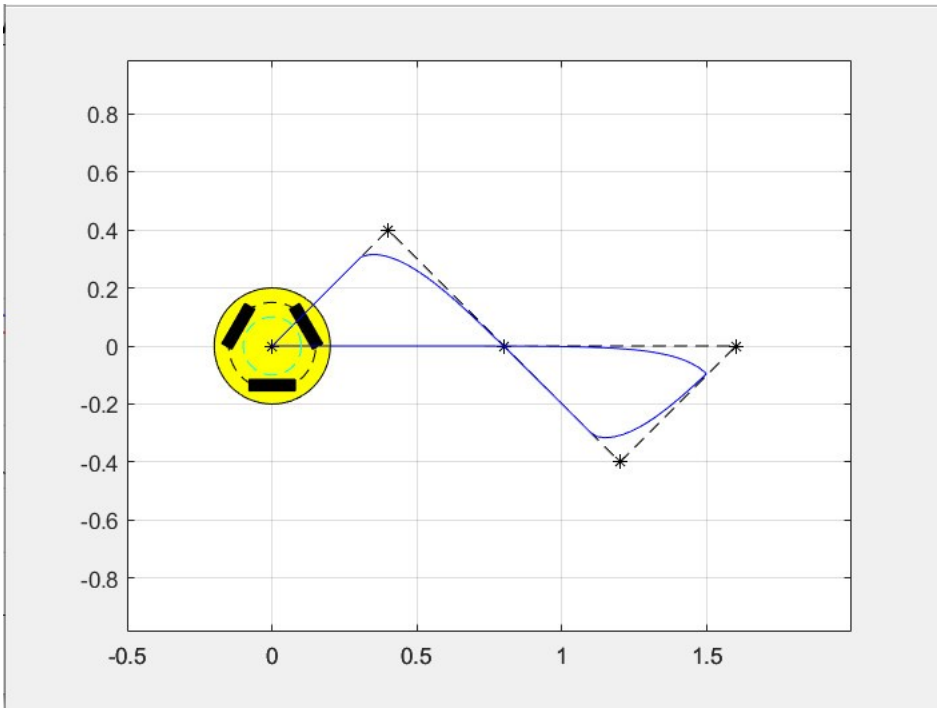


Fig. 46: *simulation of Pure Pursuit Algorithm in MATLAB*

+ Positional response over time of the vehicle during running :

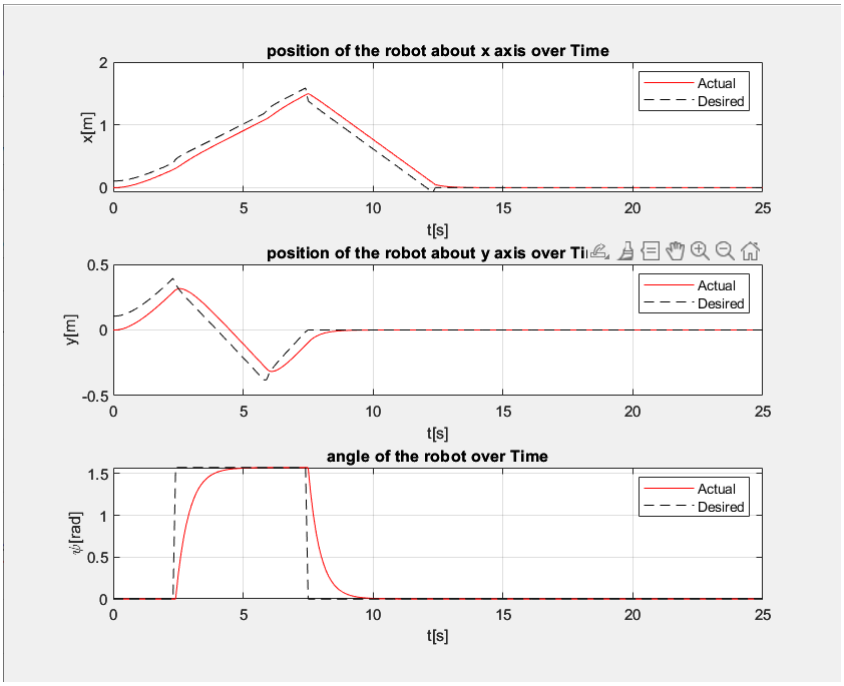


Fig. 47: *Position and pose of the robot over time*

+ The signal controls ζ over the base's timing during running :

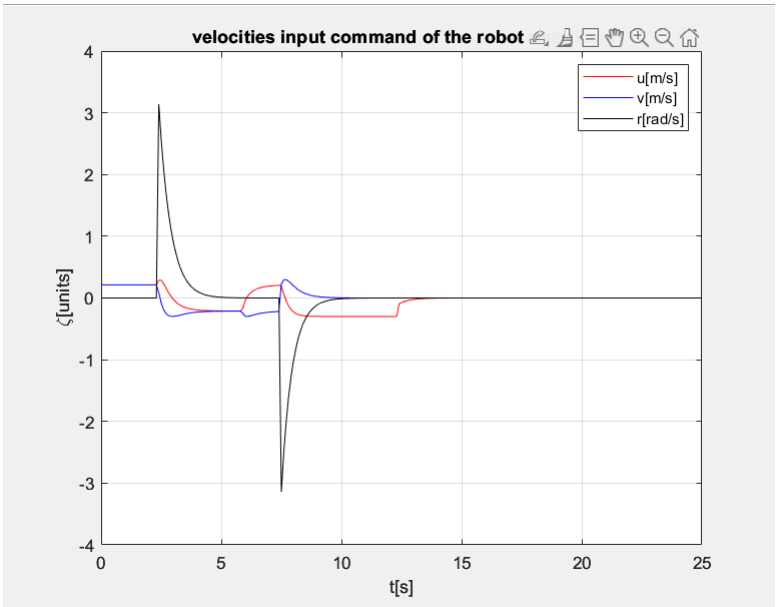


Fig. 48: `Control signal` ζ `of the robot over time`

+ Angular velocity of each wheel ω over time of the sole during running:

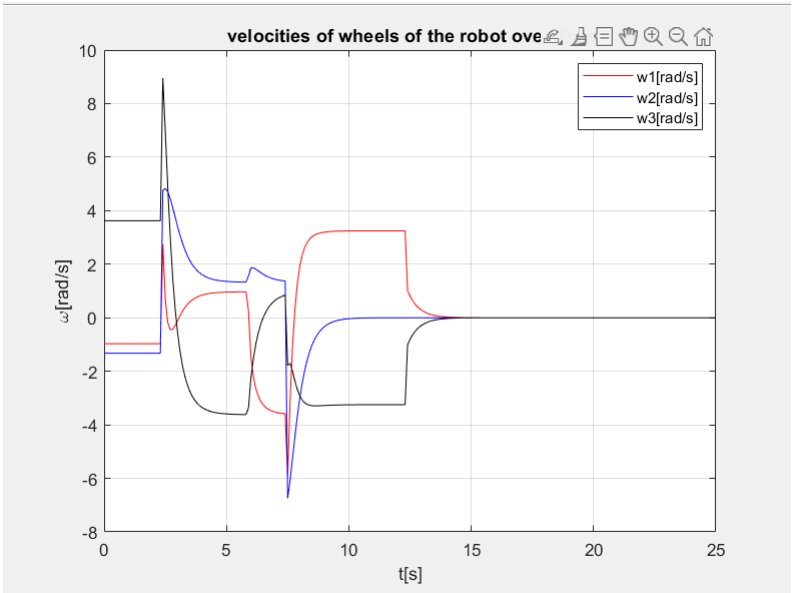


Fig. 49: *angular velocity* ω *of robot's wheels over time*

5 THEORY AND DESIGN OF INTELLIGENT ALGORITHMS

5.1 A Star Search

5.1.1 Agents and Environments

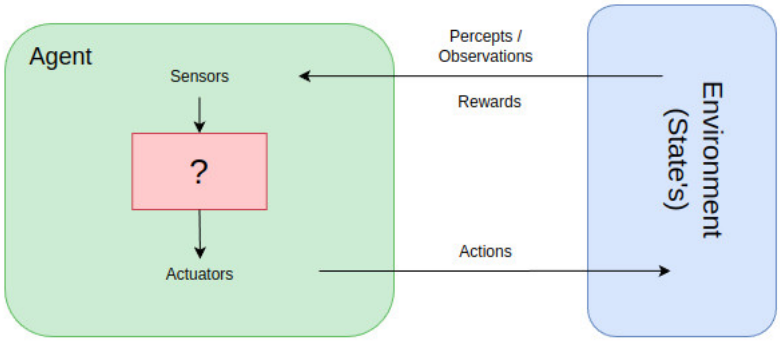


Fig. 50: Agent interacts with the environment through sensors and actuators.

An agent can be a program, software, algorithm, robot, self-driving car, or intelligent entity which can be software or hardware.

The environment is an environment in which the agent operates

An action can be executed by actuators such as motors, cylinders, etc. When an agent impacts (actions) the environment to change the environment, the environment characteristics are called state (s) and the state after being impacted are s'

Percepts/Observations are the information collected from the environment so the agent will know the current state information of the environment. The information can be collected from the agent by sensors.

In some smarter systems such as reinforcement learning to serve model training, we will add the reward system. The reward system can be generated by the environment or we can set rules to train the agent to reach the best performance.

Example: When we go to work at a company. Then the company is the environment and we are the agents. If the company wants to work hard then they will put some rules to get a higher salary. The salary in this case is the reward and we have to work hard to receive a higher salary

- Types of environments:

+ Fully observable: the agent knows completely about its current state.
+ Partially observable: the agent knows just a part about its current state.
+ Unobservable: the agent doesn't know anything about its current state.

- Exploit and explore: Explore is the agent's action to discover the environment. During the exploit, the agent gains the rewards or state of the environment

- Assumptions on the environment:

+ Fully observable: Must be able to observe the entire environment

+ Discrete: Action must be clear (turn left, turn right, ...)

+ Deterministic: Must determine the state of the environment after executing the action (Ex: the state s to s')

- Five components of a problem

+ Initial state: The starting state of the agent

+ Possible actions: The action that the agent can execute at a required state

Notation: **function** ACTIONS(state s) **return** {action1, action2, ...}

+ Transition model: the result of a state after executing the action

Notation: https://wiki.purduesigbots.com/software/control-algorithms/basic-pure-pursuit [14] **function** RESULT(state s, action a) **return** s'

+ Goal test: Determines whether the state is an incoming target or not

+ Cost: Including two types is step cost and path cost. Step cost is the cost of action. Path cost is the cost of a sequence of actions.

- Searching for Solutions

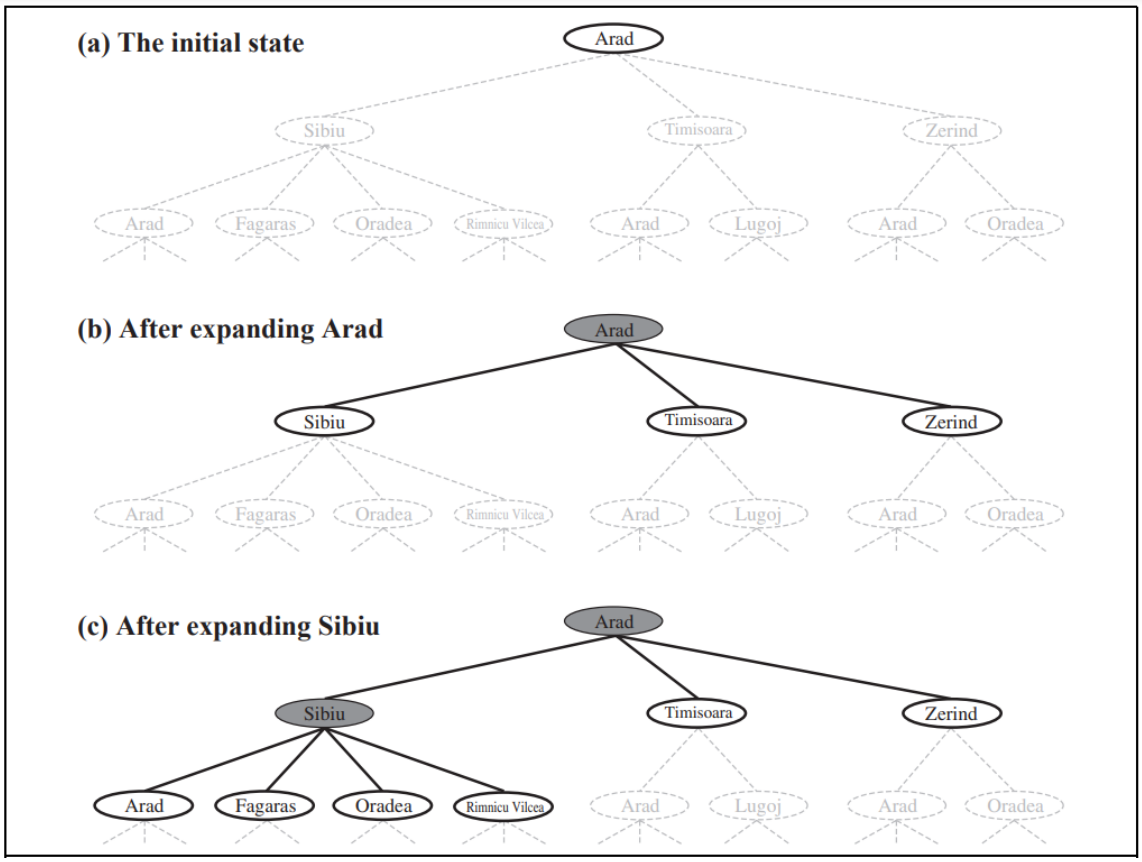


Fig. 51: Searching for finding node [15]

+ TREE SEARCH: This algorithm started with creating a set frontier containing the nodes that can expand. The disadvantage is that if it encounters a previously passed node like "Arad" in the figure above, the algorithm will continue to expand that node, causing the algorithm to fall into an endless loop.

---

function TREE-SEARCH(problem) **returns** a solution, or failure
  initialize the frontier using the initial state of the problem
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    expand the chosen node, adding the resulting nodes to the frontier

---

+ GRAPH SEARCH: This algorithm will counter the disadvantages of tree search by adding an **explored set** which contains the nodes that are explored. In the above figure, the algorithm graph search will remove "Arad" at level 2.

---

function GRAPH-SEARCH(problem) **returns** a solution, or failure
  initialize the frontier using the initial state of problem
  initialize the explored set to be empty
  loop do
    if the frontier is empty **then return** failure
    choose a leaf node and remove it from the frontier
    if the node contains a goal state **then return** the corresponding solution
    add the node to the explored set
    expand the chosen node, adding the resulting nodes to the frontier
      only if not in the frontier or explored set

- Measuring algorithms performance

+ Completeness: If there is a solution and that solution exists

+ Optimality: If the result of the returned algorithm is optimal

+ Complexity: Space (amount of RAM occupied when running). Time (time running the algorithm to find the solution)

5.1.2 Uninformed Search

Uninformed search (Blind search) is an algorithm search with no knowledge about the target they expand a node in a random way or a queue way.

- Three common variants of queues:

+ FIFO queue (first-in, first-out): the first node comes in, the first node goes out

+ LIFO queue (last-in, first-out, also call stack): the last node comes in, the first node goes out

+ Priority queue: This type is more special because adding a node will have the same priority number as the dictionary. When taken out, it will be taken in that order of priority

- Operations on queue:

+ EMPTY?(queue): return True if array empty, else return False.

+ POP(queue): get nodes according to the queue's rule order

+ INSERT(queue): insert nodes according to the queue's rule order

- Components of a node:

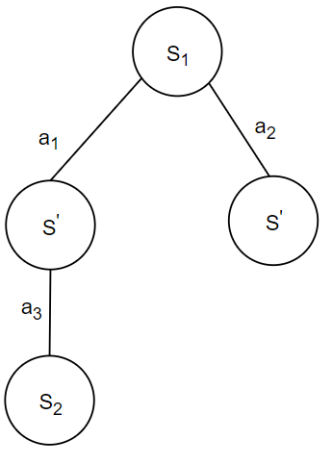


Fig. 52: Components of a node

- n.STATE: the state of a node after action (Ex: S')

- n.PARENT: the node before action to the current node (Ex: S_1)

- n.ACTION: the action can execute from the current node (Ex: a_1)

- n.PATH-COST: the cost from initial node to current node, usually denoted by g(n).

There are many algorithms for uninformed search such as breadth-first search (BFS), depth-first search (DFS), uniform cost search (UCS), etc. But to easily begin we will start with the breadth-first search and then we will see another algorithm structure of uninformed search just adjusted a little from the breadth-first search

Breadth-first search (BFS)

```
function BREADTH-FIRST-SEARCH(problem) returns a solution, or failure
  node ← a node with STATE = problem.INITIAL-STATE, PATH-COST = 0
  if problem.GOAL-TEST(node.STATE) then return SOLUTION(node)
  frontier ← a FIFO queue with node as the only element
  explored ← an empty set
  loop do
    if EMPTY?(frontier) then return failure
    node ← POP(frontier)  /* chooses the shallowest node in frontier */
    add node.STATE to explored
    for each action in problem.ACTIONS(node.STATE) do
      child ← CHILD-NODE(problem, node, action)
      if child.STATE is not in explored or frontier then
        if problem.GOAL-TEST(child.STATE) then return SOLUTION(child)
        frontier ← INSERT(child, frontier)
```

This is the basic algorithm of uniform search so in this algorithm we don't need path-cost. The input function "problem" is five components of a problem introduced above. The CHILD-NODE function works like the STATE function just append the problem variable.

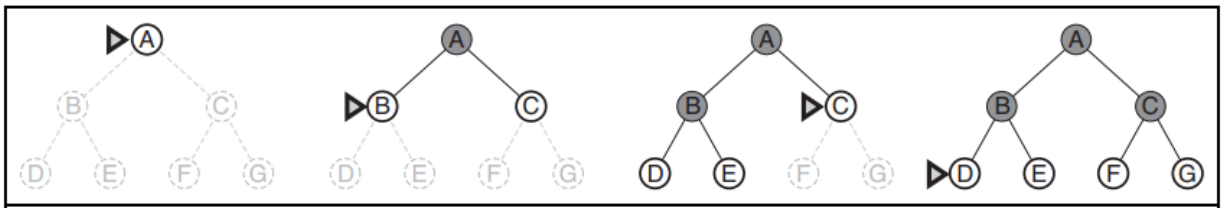


Fig. 53: Breadth-first search on a simple binary tree [15]

First, when passing the problem into the function, it will initialize the starting node. Next, we will test whether that node is the destination or not. After checking, if not, initialize the frontier array according to FIFO rules and insert the expandable nodes in order and initialize the explored array as an empty set. Then enter the loop, start the loop to check if that frontier is empty, if empty then return False. Next, if the check is not empty, we will take the node from the frontier array according to FIFO rules and add that node to the explore array (to avoid encountering that node again and falling into an endless loop). Next, we will perform the actions that the node can perform and then check if that node is in the explore and frontier arrays. If not, continue to check if it is the target node we want to go to. If true, return the solution, otherwise continue adding nodes created from action to the frontier array and repeat the loop until a solution is found.

5.1.3 Informed Search

As we have seen, uninformed search selects nodes according to specific rules without any information about the target. In some cases, it will be a waste of time and storage. The information search was born to overcome that waste. By adding heuristic values to the calculation functions, the search process prioritizes that node more effectively. It's the same as the uninformed cost search but more effective when adding a heuristic function to make it reach the target point. much faster.

Evaluation function

This function will decide the priority of nodes. If the uniform cost function has a priority function f(n) = n.PATH-COST, then inform search will add the heuristic h(n) to the function f(n), meaning f(n) = n.PATH-COST + h(n)

Heuristic function

This function is used to estimate the cost from the current node to the target node. Usually denoted by h(n).

The range in which h(n) operates makes information search effective is [0, actual cost].

If your function h(n) has a calculated value greater than the actual cost, it will not work well anymore. If h(n) equals 0 for all n then f(n) = n.PATH-COST means this function will be the uninformed cost search.

In actuality, we must find the function h(n) whose value is as close to the actual value as possible.

5.1.4 A Star search

---

function A-STAR-SEARCH(problem) **returns** a solution, or failure
  node ←a node with STATE = *problem*.INITIAL-STATE, PATH-COST = 0
  $frontier$ ←a priority queue order by f(n), with *node* as the only element
  explored ←an empty set
  loop do
    if EMPTY?(frontier) **then return** failure
    node ←POP(frontier)
    if *problem*.GOAL-TEST(node.STATE) **then return** SOLUTION(node)
    add node.STATE to *explored*
    for each action in *problem*.ACTIONS(node.STATE) **do**
      child ←CHILD-NODE(problem,node,action)
      if *child*.STATE is not in *explored* or $frontier$ **then**
        $frontier$ ←INSERT(*child*.$frontier$)
      else if*child*.STATE is in $frontier$ with higher f(n) **then**
        replace that $frontier$ node with *child*

This algorithm selects nodes based on the value of the function f(n). Nodes with lower f(n) values are given priority

The function formula f(n) is calculated f(n) = g(n) + h(n) with g(n) being the actual value.

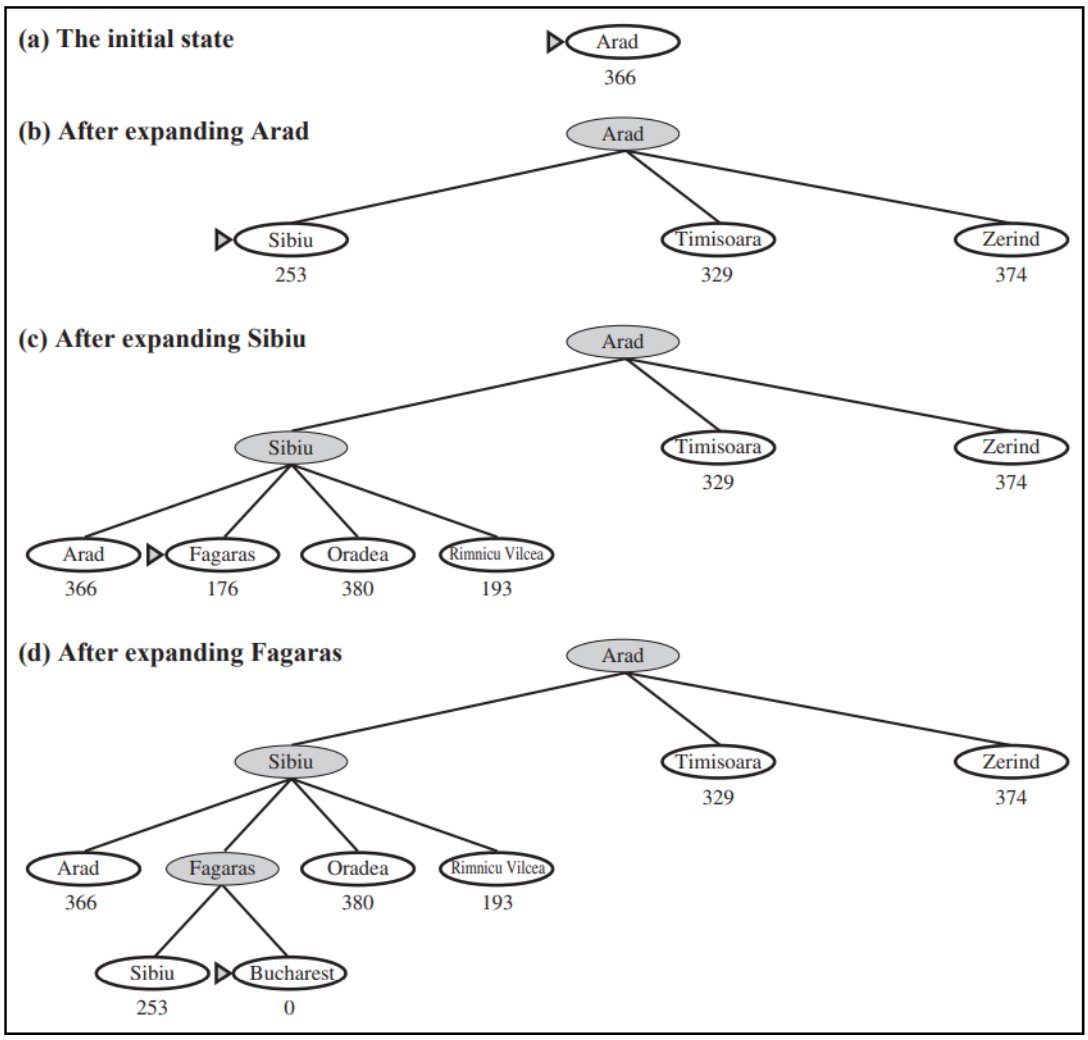


Fig. 54: A-star search on a simple binary tree [15]

First, we will determine the state of the starting node, then we will initialize the frontier array and add expandable nodes with priority order (here calculated according to the function f(n)). Next, we will initialize the explored array to contain the explored nodes (to avoid falling into an endless loop). After initialization, the loop will enter. And start checking if the frontier array is empty. If empty, return failure. If not empty, that node will be removed from the frontier with the priority with the lowest f(n) value. Then check to see if it is the target node you want to go to, if so, print out the solution. Next, add that node to the explore array. Next, perform the actions that the node can perform. Then check the nodes that are discovered after performing that action. If they are not in the frontier or discovered array, they will be added to the frontier, but if they are in the frontier array with a higher f(n) value, they will be replaced. So the frontier value is in that array. Repeat the loop until the target point is found.

Advantage:

Find the shortest path. Searching is faster than the uninformed search

Disadvantage:

Space complexity because it must contain node information in both width and depth, so it also requires a lot of memory

5.2 Natural Language Processing

5.2.1 Theory of natural language processing and Chatbots

Natural Language Processing (NLP) is a field of artificial intelligence (AI) focused on helping computers understand, evaluate, and interact with natural human language. The main goal of NLP is to help computers process, understand and produce natural language effectively, similar to the way humans interact with each other through speech or text.

In the field of NLP, a popular application is chatbots. A chatbot is a computer program designed to automatically interact with users through chat channels such as websites, mobile applications or social media platforms. Chatbots use NLP to naturally understand and respond to messages from users, helping them find information, answer questions, or even perform specific tasks that were previously only available to humans. can be done.

NLP and chatbots are becoming an important part of many modern technology applications, from automated translation and natural language processing to better user experiences in mobile applications and websites. Progress in this field improves the interaction between computers and humans, bringing convenience and flexibility to users.

• **Types of Chatbos**:

+ Rule-Based Chatbots: Rule-based chatbots, also known as decision-tree bots, operate on a set of predetermined rules. These rules serve as the foundation for the range of issues the chatbot can handle and the corresponding solutions it can provide. Similar to a flowchart, rule-based chatbots outline the flow of conversations.

+ Conversational AI Chatbots: AI-enabled chatbots leverage the capabilities of Machine Learning (ML) and Natural Language Processing (NLP) to comprehend the context and intent behind a question before generating a response.

+ Contextual Chatbots: To retain the history of previous conversations, these chatbots employ Machine Learning (ML) and Artificial Intelligence (AI) technologies, learning and evolving over time, especially in relation to specific user s.

+ Voice-Enabled Chatbots: Voice-enabled chatbots process user inputs through spoken language, execute requests, respond to queries, and perform various creative tasks. Through the implementation of text-to-speech (TTS) and voice recognition APIs, businesses can create their own voice-activated chatbot.

+ Hybrid Chatbots: Hybrid chatbots combine rule-based and AI-based approaches, offering a balanced solution with the simplicity of rule-based bots and the contextual understanding of AI bots for effective customer interactions.

Comment: With the need for hardware performance and quick response, I chose to use a Rule-Based Chatbot, with the ability to combine agility and high accuracy.

5.2.2 Overview of Chatbot diagram

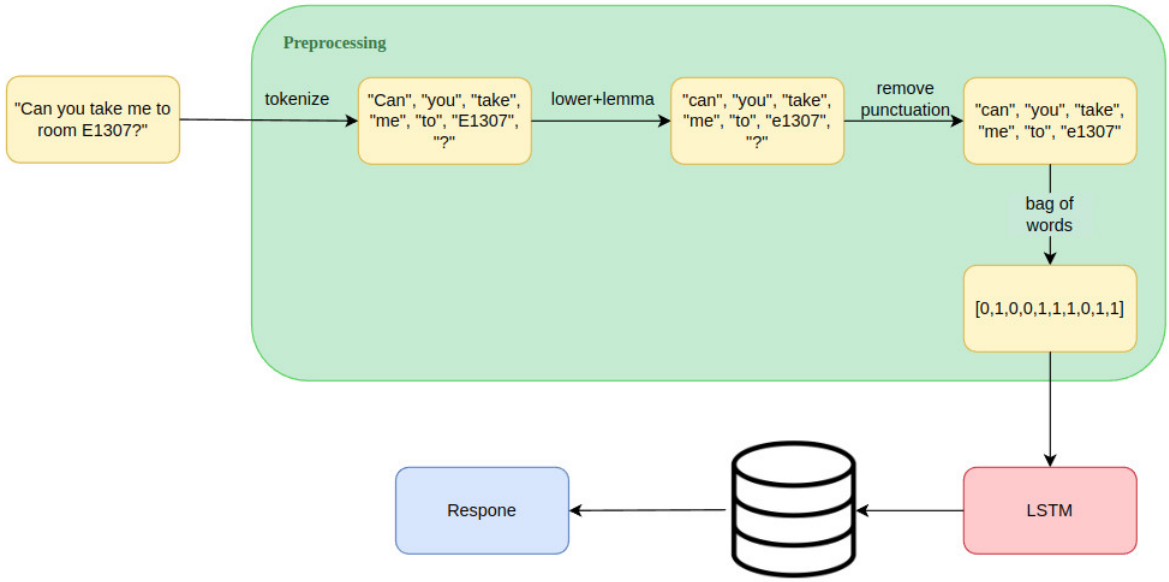


Fig. 55: Pipeline Chatbot

First, the user enters text. Then through the tokenizer layer, the words in the sentence will be separated and then through the lower and lemma layers to bring back the original word and turn the uppercase word into lowercase. And the model cannot understand sesame words so we need to convert it to digital form using the bag of words method. Then the model will guess to which class this statement belongs to and from there access the database to give the corresponding response.

5.2.3 Overview of text preprocessing

- Tokenization: the text is split into smaller units. We can use either sentence tokenization or word tokenization based on our problem statement.

msg_lower	msg_tokenied
go until jurong point crazy available only in bugis n great world la e buffet cine there got amore wat	[go, until, jurong, point, crazy, available, only, in, bugis, n, great, world, la, e, buffet, cine, there, got, amore, wat]
ok lar joking wif u oni	[ok, lar, joking, wif, u, oni]
free entry in 2 a wkly comp to win fa cup final tkts 21st may 2005 text fa to 87121 to receive entry questionstd txt ratetcs apply 08452810075over18s	[free, entry, in, 2, a, wkly, comp, to, win, fa, cup, final, tkts, 21st, may, 2005, text, fa, to, 87121, to, receive, entry, questionstd, txt, ratetcs, apply, 08452810075over18s]
u dun say so early hor u c already then say	[u, dun, say, so, early, hor, u, c, already, then, say]
nah i dont think he goes to usf he lives around here though	[nah, i, dont, think, he, goes, to, usf, he, lives, around, here, though]

Fig. 56: Tokenize work [16]

- Stop Word Removal: Stopwords are the commonly used words and are removed from the text as they do not add any value to the analysis. These words carry less or no meaning. In NLTK library consists of a list of words that are considered stopwords for the English language. Some of them are : [i, me, my, myself, we, our, ours, ourselves, you, you're, you've, you'll, you'd, your, yours, yourself, yourselves, he, most, other, some, such, no, nor, not, only, own, same, so, then, too, very, s, t, can, will, just, don, don't, should, should've, now, d, ll, m, o, re, ve, y, ain, aren't, could, couldn't, didn't, didn't]

msg_tokenied	no_stopwords
[go, until, jurong, point, crazy, vailable, only, in, bugis, n, great, world, la, e, buffet, cine, there, got, amore, wat]	[go, jurong, point, crazy, available, bugis, n, great, world, la, e, buffet, cine, got, amore, wat]
[ok, lar, joking, wif, u, oni]	[ok, lar, joking, wif, u, oni]
[free, entry, in, 2, a, wkly, comp, to, win, fa, cup, final, tkts, 21st, nay, 2005, text, fa, to, 87121, to, receive, entry, questionstd, txt, ratetcs, apply, 08452810075over18s]	[free, entry, 2, wkly, comp, win, fa, cup, final, tkts, 21st, may, 2005, text, fa, 87121, receive, entry, questionstd, txt, ratetcs, apply, 08452810075over18s]
[u, dun, say, so, early, hor, u, c, already, then, say]	[u, dun, say, early, hor, u, c, already, say]
[nah, i, dont, think, he, goes, to, usf, he, lives, around, here, though]	[nah, dont, think, goes, usf, lives, around, though]

Fig. 57: Stop Word Removal work [16]

- Stemming: also known as the text standardization step where the words are stemmed or diminished to their root/base form. The disadvantage of stemming is that it stems the words such that its root form loses the meaning or it is not diminished to a proper English word. We will see this in the steps done below.

no_stopwords	msg_stemmed
[go, jurong, point, crazy, available, bugis, n, great, world, la, e, buffet, cine, got, amore, wat]	[go, jurong, point, crazi, avail, bugi, n, great, world, la, e, buffet, cine, got, amor, wat]
[ok, lar, joking, wif, u, oni]	[ok, lar, joke, wif, u, oni]
[free, entry, 2, wkly, comp, win, fa, cup, final, tkts, 21st, may, 2005, text, fa, 87121, receive, entry, questionstd, txt, ratetcs, apply, 08452810075over18s]	[free, entri, 2, wkli, comp, win, fa, cup, final, tkt, 21st, may, 2005, text, fa, 87121, receiv, entri, questionstd, txt, ratetc, appli, 08452810075over18]
[u, dun, say, early, hor, u, c, already, say]	[u, dun, say, earli, hor, u, c, alreadi, say]
[nah, dont, think, goes, usf, lives, around, though]	[nah, dont, think, goe, usf, live, around, though]

Fig. 58: Stemming work [16]

- Lemmatization: It stems the word but makes sure that it does not lose its meaning. Lemmatization has a pre-defined dictionary that stores the context of words and checks the word in the dictionary while diminishing. The difference between Stemming and lematization can be understood with the example provided below.

no_stopwords	msg_stemmed	msg_lemmatized
[go, jurong, point, crazy, available, bugis, n, great, world, la, e, buffet, cine, got, amore, wat]	[go, jurong, point, crazi, avail, bugi, n, great, world, la, e, buffet, cine, got, amor, wat]	[go, jurong, point, crazy, available, bugis, n, great, world, la, e, buffet, cine, got, amore, wat]
[ok, lar, joking, wif, u, oni]	[ok, lar, joke, wif, u, oni]	[ok, lar, joking, wif, u, oni]
[free, entry, 2, wkly, comp, win, fa, cup, final, tkts, 21st, may, 2005, text, fa, 87121, receive, entry, questionstd, txt, ratetcs, apply, 08452810075over18s]	[free, entri, 2, wkli, comp, win, fa, cup, final, tkt, 21st, may, 2005, text, fa, 87121, receiv, entri, questionstd, txt, ratetc, appli, 08452810075over18]	[free, entry, 2, wkly, comp, win, fa, cup, final, tkts, 21st, may, 2005, text, fa, 87121, receive, entry, questionstd, txt, ratetcs, apply, 08452810075over18s]
[u, dun, say, early, hor, u, c, already, say]	[u, dun, say, earli, hor, u, c, alreadi, say]	[u, dun, say, early, hor, u, c, already, say]
[nah, dont, think, goes, usf, lives, around, though]	[nah, dont, think, goe, usf, live, around, though]	[nah, dont, think, go, usf, life, around, though]

Fig. 59: Comparing between Stemming and Lemmatization [16]

- Bag of words: It's an algorithm that transforms the text into fixed-length vectors. This is possible by counting the number of times the word is present in a document. The word occurrences allow to compare different documents and evaluate their similarities for applications, such as search, document classification, and topic modeling.

	it	is	puppy	cat	pen	a	this
it is a puppy	1	1	1	0	0	1	0
it is a kitten	1	1	0	0	0	1	0
it is a cat	1	1	0	1	0	1	0
that is a dog and this is a pen	0	2	0	0	1	2	1
it is a matrix	1	1	0	0	0	1	0

Fig. 60: How bag of work works [16]

5.2.4 Recurrent Neural Networks and Long Short-Term Memory

• Recurrent Neural Networks (RNN)

In deep learning, we have two big models, the first is the Convolutional Neural Network (CNN) for process image, and the second is the Recurrent Neural Network (RNN) for process sequence. In this problem, the problem we need to handle is text, which means processing it as a sequence, so we will focus on RNN.

RNNs are capable of retaining information from previous steps in sequence processing, which helps them understand context and time dependencies. The basic idea of RNN is to transfer information from the previous time step into the current time step.

Application of RNN: Speech-to-text, Sentiment classification, Machine translation, etc.

- Classification of RNN problems

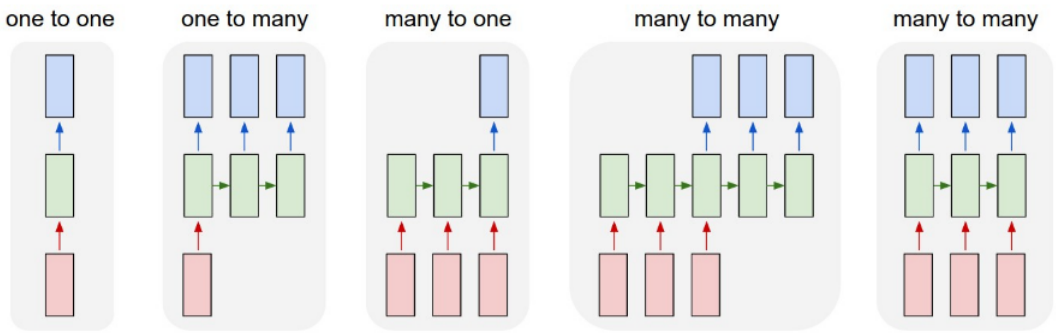


Fig. 61: Classification of RNN problems [17]

+ **One to one:** 1 input and 1 output, for example with CNN input is image and output is segmented image.

+ **One to many:** the problem has 1 input but many outputs, for example: the photo caption problem, the input is 1 photo but the output is many words describing that photo, in the form of a sentence.

+ **Many to one:** the problem has many inputs but only 1 output, for example the problem of classifying actions in a video, the input is many images (frames) separated from the video, output is the action in the video.

+ **Many to many:** the problem has many inputs and many outputs, for example, the problem of translating from English to Vietnamese, the input is a sentence with many words: "I love Vietnam" and the output is also a sentence with many words "I love Vietnam".

- RNN model for the problem.

The model has 10 inputs and 1 output, corresponding to the input order of the bag of words size

Each circle is called a state, state t has input x_t and s_{t-1} (output of previous state); the output is $s_t = f(U * x_t + W * s_{t-1})$. $f(...)$ is the activation function usually Tanh or ReLU.

We can see s_t bring all information from the previous state (s_{t-1}) and input of current state $(x_t) \Rightarrow x_t$ is like memory that remembers the characteristics of word inputs from x_1 to x_t.

When started, there is no input, so S_0 equals 0.

Because in this problem we just have one output, so we just take the output of the last state. Then S_{10} will learn the information from all inputs: $\hat{y} = g(V * s_{10})$. With $g(...)$ is a softmax function because this is a classification problem.

- Loss function

The loss function of the whole model is equal to the total loss of each output, however in the above model there is only 1 output, and is a classification problem so categorical cross-entropy loss will be used.

$$L = -\sum_{i=1}^{10} y_i * log(\hat{y}_i)$$

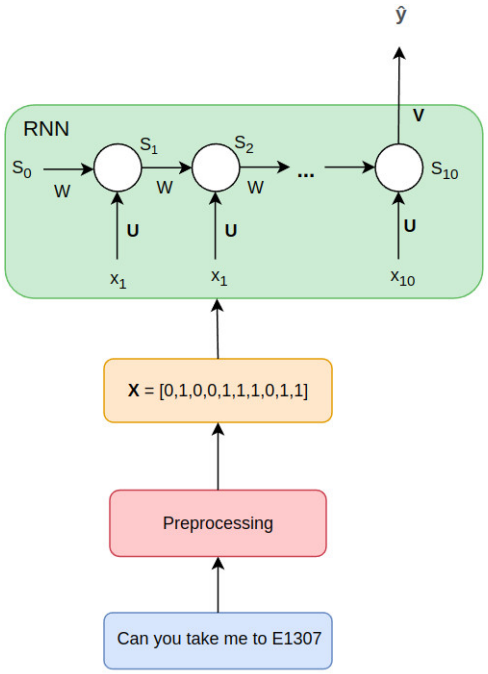


Fig. 62: RNN model for the problem [17]

- Activation Functions

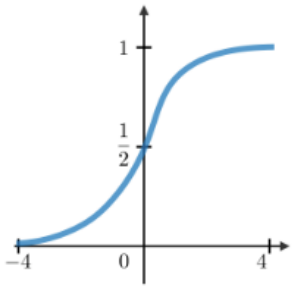
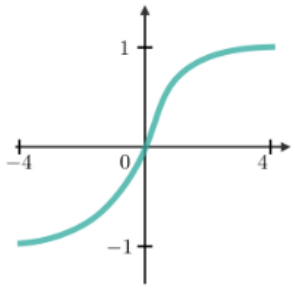
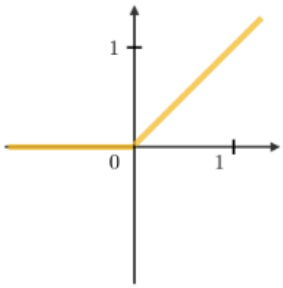
Sigmoid	Tanh	RELU
$g(z) = \frac{1}{1+e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$
		

Fig. 63: Activation Functions [17]

- Backpropagation Through Time

We have three parameters to find that is V, U, W. To execute gradient descent, we have to find $\frac{\partial L}{\partial V}, \frac{\partial L}{\partial U}, \frac{\partial L}{\partial W}$

With derivative of V:

$$\frac{\partial L}{\partial V} = \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial V}$$

With derivative of U:

$$\frac{\partial L}{\partial U} = \frac{\partial L}{\partial \hat{y}} * \frac{\hat{y}}{\partial s_{10}} * \frac{\partial s_{10}}{\partial U}$$

With derivative of W:

$$\frac{\partial L}{\partial U} = \frac{\partial L}{\partial \hat{y}} * \frac{\hat{y}}{\partial s_{10}} * \frac{\partial s_{10}}{\partial W}$$

- Two issues of Standard RNN

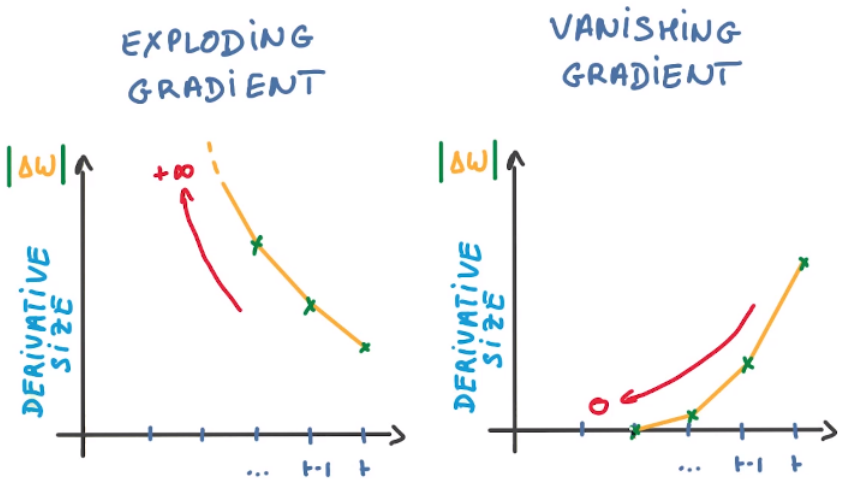


Fig. 64: Two issues of Standard RNN [17]

+ Exploding Gradient: This phenomenon occurs when weights suddenly increase for no reason

+ Vanishing Gradient: This phenomenon occurs when the derivative value is too small, making the model unable to be updated

- Advantages and disadvantages of RNN

+ Advantages: Handle sequential data, share weights across time steps, and process inputs of any length.

+ Disadvantages: Easy to occur vanishing and exploding gradient, limited ability to handle long strings.

• **Long Short-Term Memory (LSTM)**

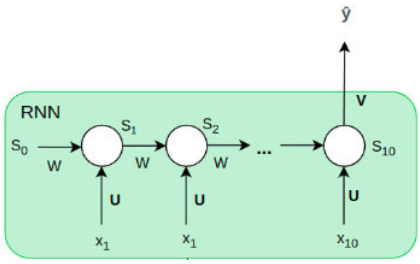


Fig. 65: RNN model

We have $\frac{\partial L}{\partial L} = \frac{\partial L}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial s_{10}} * \frac{\partial s_{10}}{\partial s_i} * \frac{\partial s'}{\partial W}$. With $\frac{\partial S_{10}}{\partial S_i} = \prod_{j=i}^{9} \frac{\partial s_{j+1}}{\partial s_j}$ [18]

Suppose the activation function is Tanh, so We have $s_t = tanh(U * x_t + W * s_{t-1})$

$\Rightarrow \frac{\partial s_t}{\partial s_{t-1}} = W * (1 - s_t^2)$

Because limit of Tanh is (-1,1), so we have $s_j < 1 \Rightarrow$ In distant states $\frac{\partial s_{10}}{\partial s_i} \approx 0 \Rightarrow$ Occur vanishing gradient.

So in theory, RNN can carry information from previous layers to the following layers, but in reality, information can only be carried through a certain number of states, after which the vanishing gradient will occur, or in other words is a model that only learns from states near it $\Rightarrow$ **short term memory.**

Let's try to take an example of short term memory. The problem is to predict the next word in the passage. In the first paragraph "The sun rises in the direction of ...", we can only use the previous words in the sentence to guess east. However, with the paragraph, "I am Vietnamese. I am living abroad. I can speak fluently in ..." then it is clear that just using the word in that sentence or the previous sentence cannot predict that the word to fill in is Vietnamese. We need information from a very far previous state $\Rightarrow$ need **long term** memory which RNN cannot do $\Rightarrow$ Need a new model to solve this problem $\Rightarrow$ **Long short term memory (LSTM)** was born.

- LSTM model

At state t of LSTM model we have:

+ Input: c_{t-1}, h_{t-1}, x_t. With x_t is the input at sate t. c_{t-1}, h_{t-1} is the output of the previous layer. h plays a role quite similar to s in RNN, while c is a new point of LSTM.

+ Output: c_t, h_t we call c_t is cell state, h_t is hidden state.

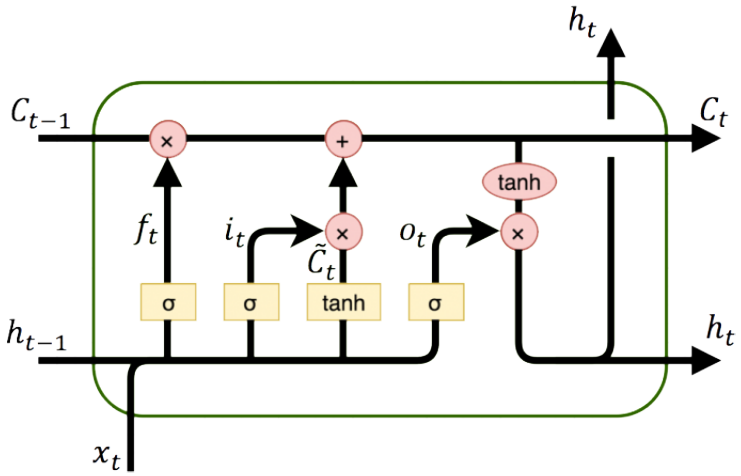


Fig. 66: LSTM model [18]

+ Forget gate: $f_t = \sigma(U_f * x_t + W_f * h_{t-1} + b_f)$

+ Input gate: $i_t = \sigma(U_i * x_t + W_i * h_{t-1} + b_i)$

+ Output gate: $o_t = \sigma(U_o * x_t + W_i * h_{t-1} + b_o)$

Comment: $0 < f_t, i_t, o_t < 1$; b_f, b_i, b_o are the bias coefficients, the coefficients $W$and U are the same as in the RNN model.

+ $\tilde{c_t} = tanh(U_c * x_t + W_c * h_{t-1} + b_c)$, this step is similar to calculating s_t in RNN.

+ $c_t = f_t * c_{t-1} + i_t * \tilde{c_t}$, the forget gate decides how much to take from the previous cell state and the input gate decides how much to take from the input of the state and hidden layer of the previous layer.

+ $h_t = o_t * tanh(c_t)$, the output gate decides how much needs to be taken from the cell state to become the output of the hidden state.

Comment: because we have $h_t, \tilde{c_t}$ quite similar to RNN, so the model has **short term memory**. While c_t is like a conveyor belt on the RNN model, any information that is important and used later will be sent in and used when needed ⇒ can carry information from far away ⇒ **long term memory**. Therefore, the LSTM model has both short term memory and long term memory.

- LSTM resists vanishing gradients

Because there is an additional c_t that stores old value information, vanishing gradients can be avoided.

5.3 Database

```
Database changed
MariaDB [tensorbot]> select * from target_dict;
+----+-----------+------------+---------------------+
| id | target    | coordinate | info                |
+----+-----------+------------+---------------------+
|  1 | Tensorbot | [4, 10]    | NULL                |
|  2 | Quang     | [4, 10]    | NULL                |
|  3 | E1304     | [2,4]      | NULL                |
|  4 | Triet     | [1,2]      | NULL                |
|  5 | Home      | [0, 0]     | NULL                |
|  6 | E1310     | [4,10]     | meeting room        |
|  7 | E1307     | [0,0]      | servo practice room |
+----+-----------+------------+---------------------+
7 rows in set (0.00 sec)
```

Fig. 67: Database

- **id:** This is the primary key of the table, responsible for uniquely identifying each row in the table.

- **target:** This column stores the target name.

- **coordinate:** This column stores the coordinates of the target, in array form.

- **info:** This column is to store detailed information about the target.

5.4 Main algorithm flowchart

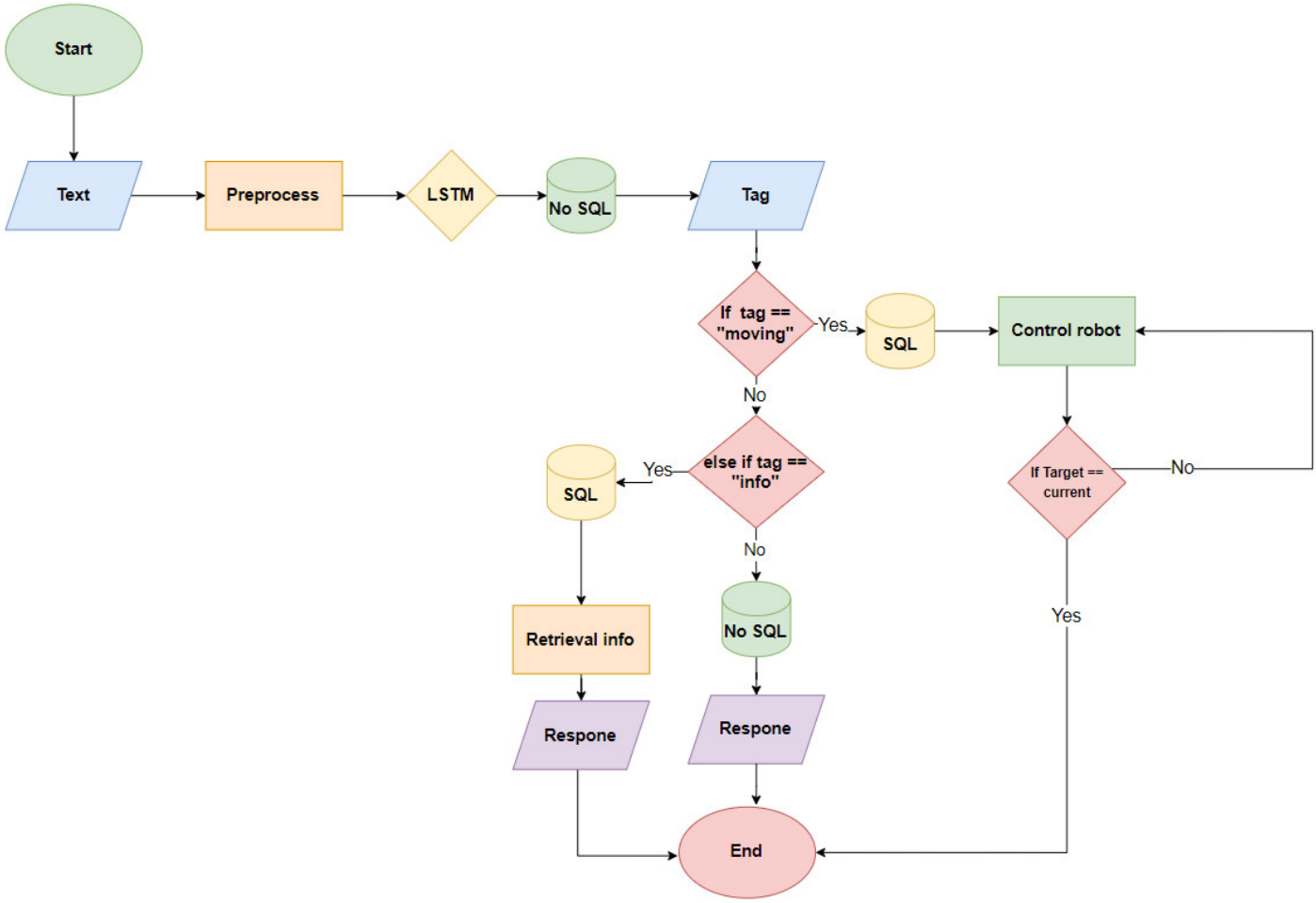


Fig. 68: Robot overview block diagram

First, when the text is entered, it will go through a preprocessing layer to put it into the model. After going through the model, the label will be output in the form of a number and from that number query the no sql database and give the tag. After issuing tags, depending on each tag, a corresponding action will be given. If the tag is "moving" it will tell the robot to act and then respond. If the tag is "info" it will query the sql database and then give the information from the object that you need to find and the information contained in the corresponding database. If there are no more, a normal response is given and the function ends

6 EXPERIMENTAL RESULTS

6.1 Experimental results of the point-to-point algorithm on robotino

- Experiment with real vehicles running in room E1303 :

+ We start do the experiment with set point $\eta_d[1, 1, 1]$

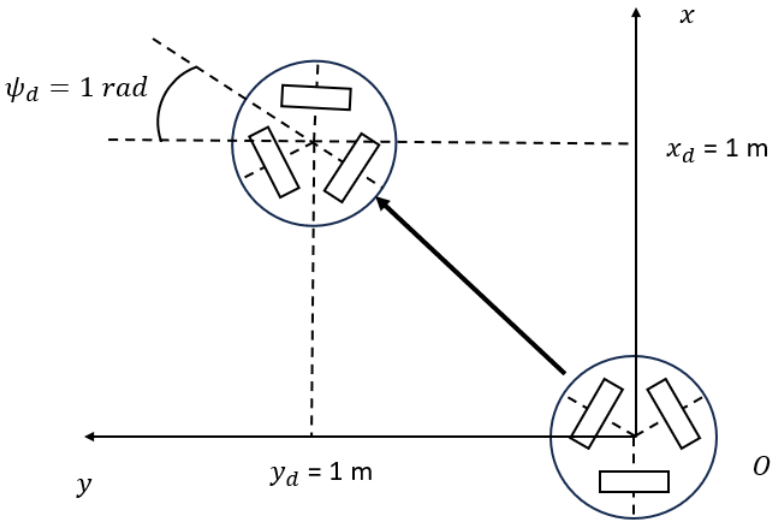


Fig. 69: *Analysis of Point-to-point algorimth of setpoint* $\eta[1, 1, 1]$

Fig. 70: *Real image of robotino when applying point-to-point algorithm*

+ We start do the experiment with set point $\eta_d[1, 1, 1]$

+ We then collect device response data over time

- Analysis the vehicle's response :

+ To the set point :

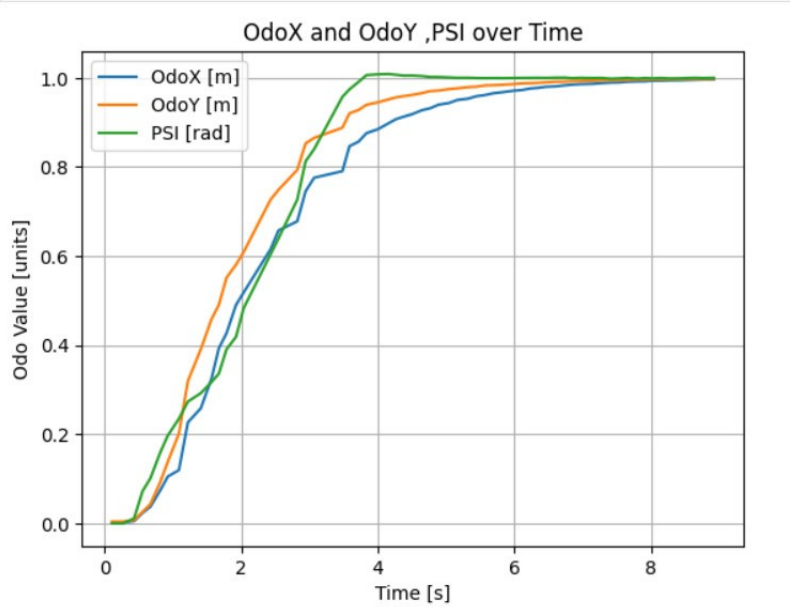


Fig. 71: *Position response of vehicle over time*

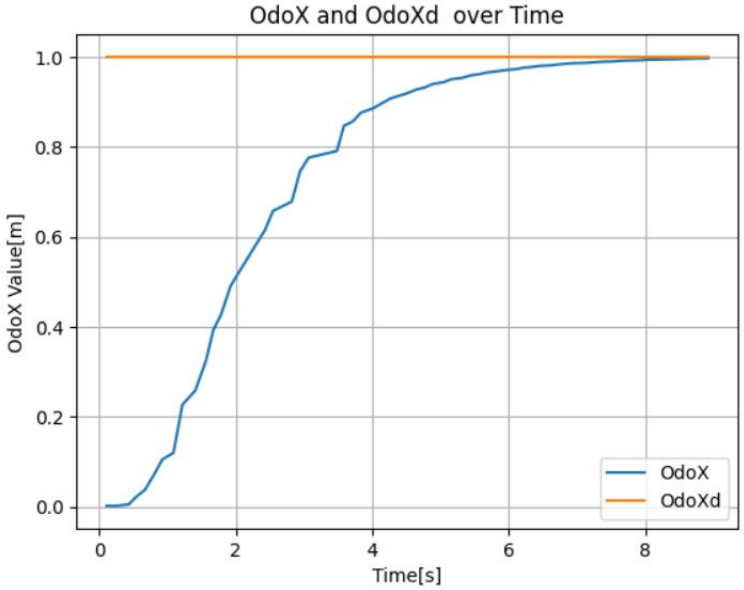


Fig. 72: *X axis response of vehicle over time*

+ From Setpoint back to home:

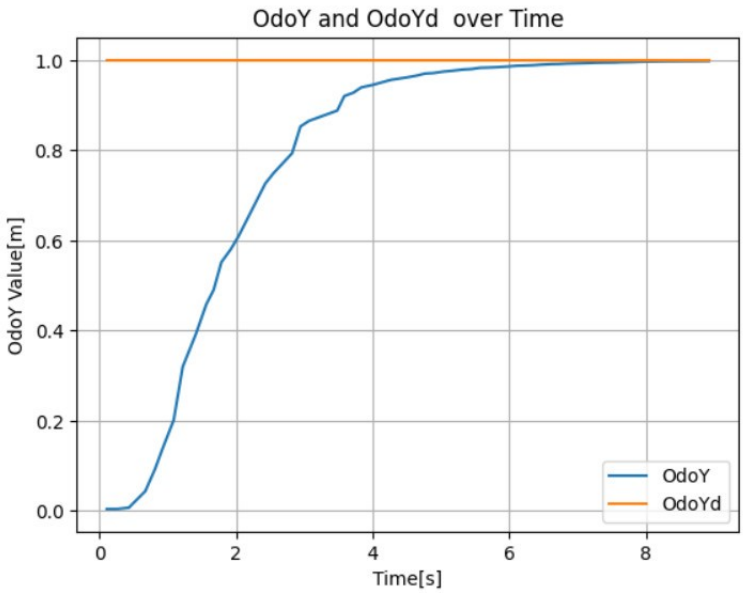


Fig. 73: *Y axis response of vehicle over time*

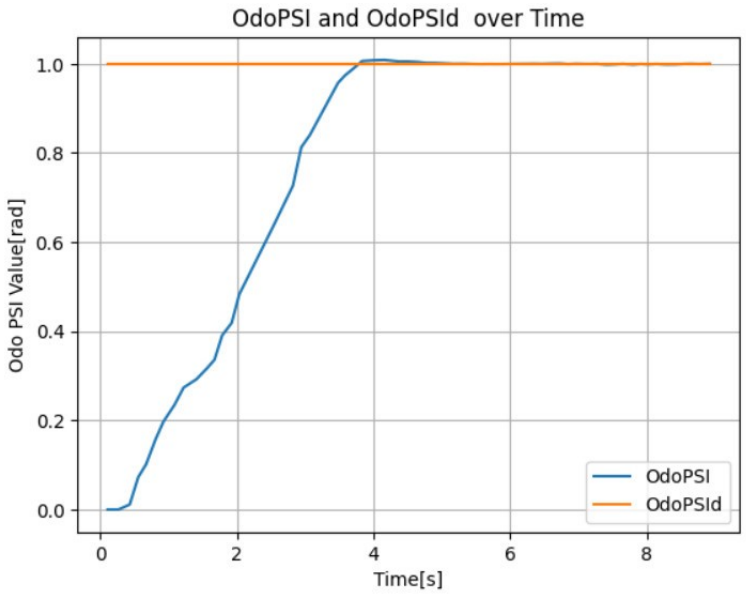


Fig. 74: *Angle response of vehicle over time*

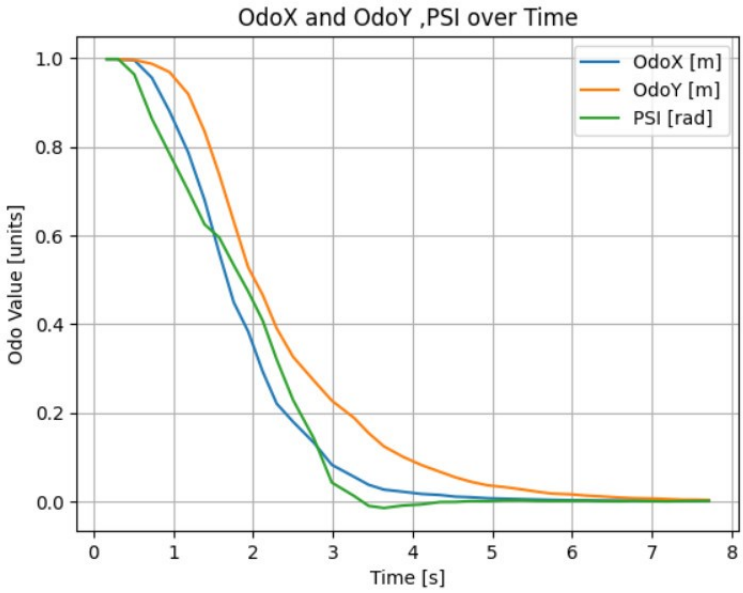


Fig. 75: *Position response of vehicle over time*

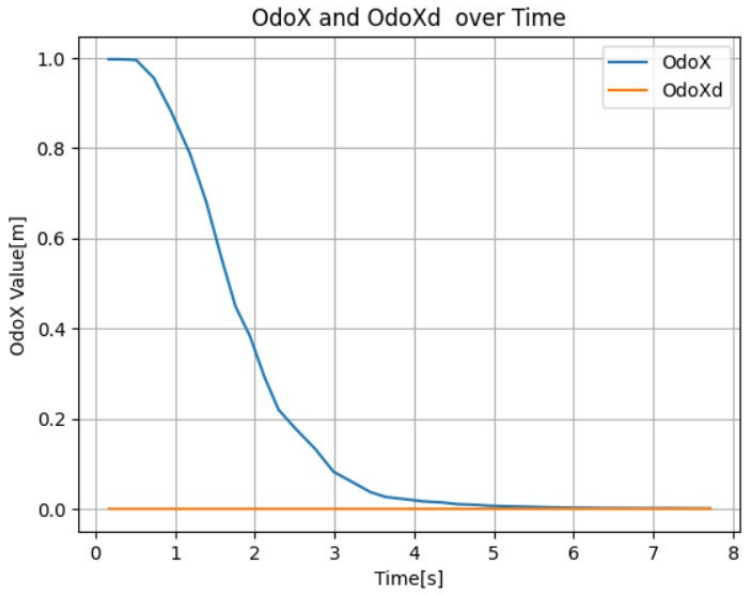


Fig. 76: *X axis position response of vehicle over time*

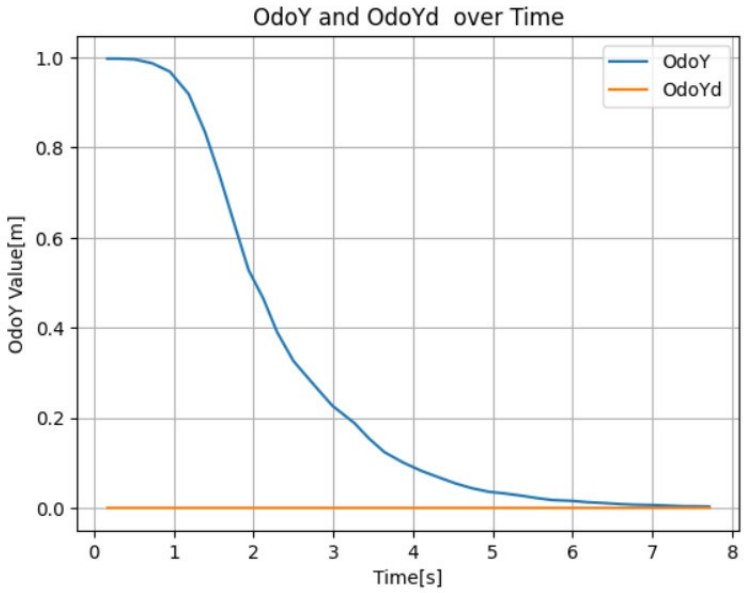


Fig. 77: *Y axis position response of vehicle over time*

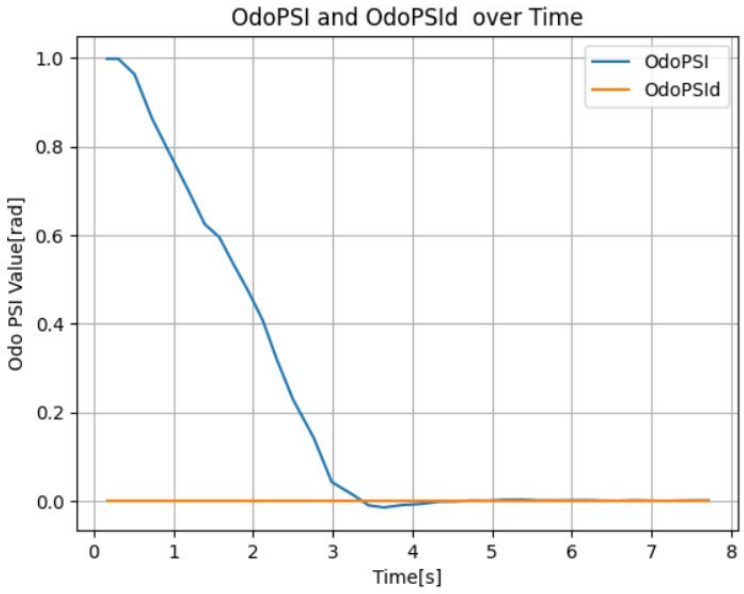


Fig. 78: *Angle response of vehicle over time*

6.2 Experimental results of the Pure pursuit algorithm on Robotino

- Experiment with real vehicles running in room E1303 :

+ We start give path to test the algorithm as follow :

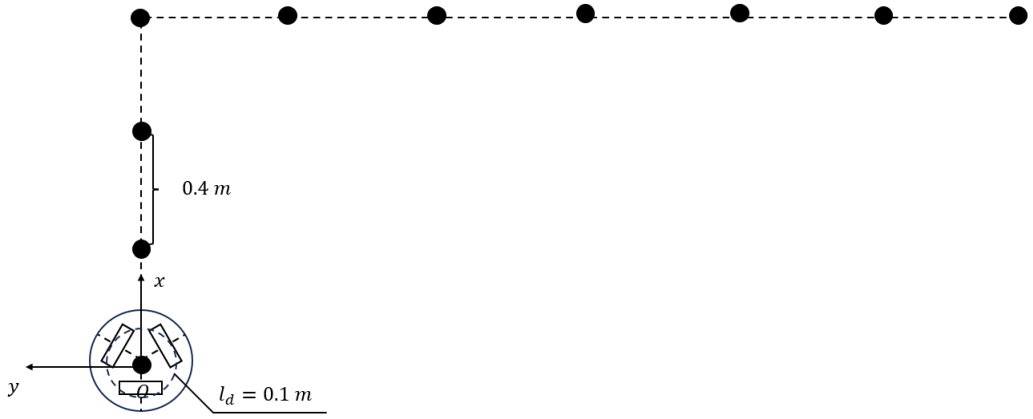


Fig. 79: *Analysis of path tracking experiment on Robotino*

- Analysis the vehicle's response :

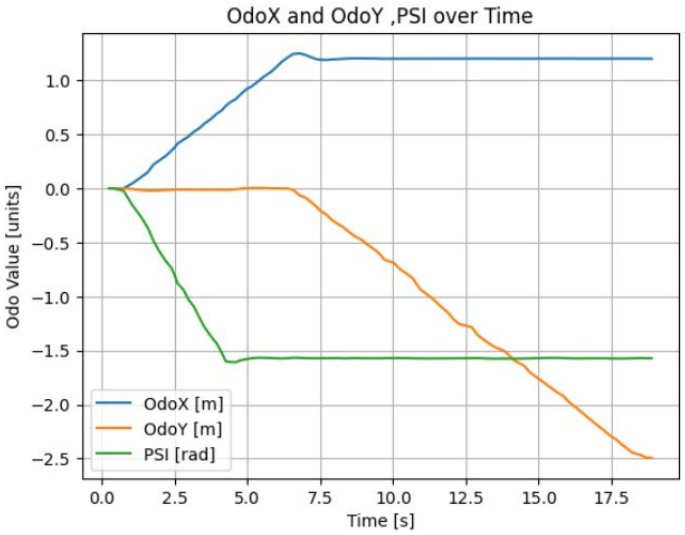


Fig. 80: *Position response of vehicle over time*

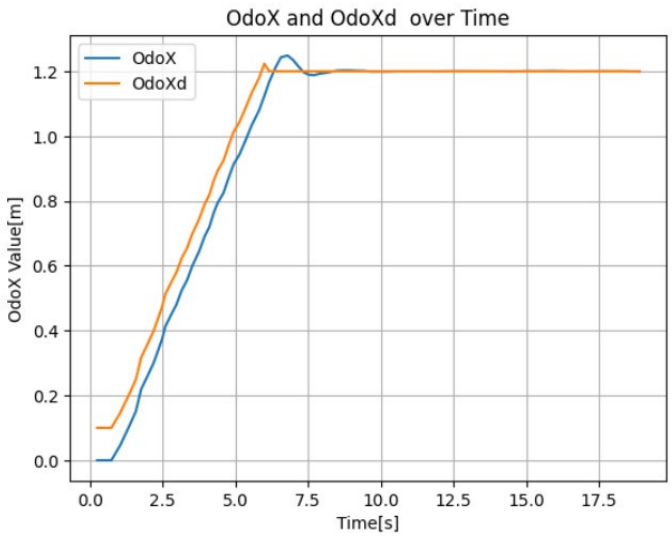


Fig. 81: *X axis position response of vehicle over time*

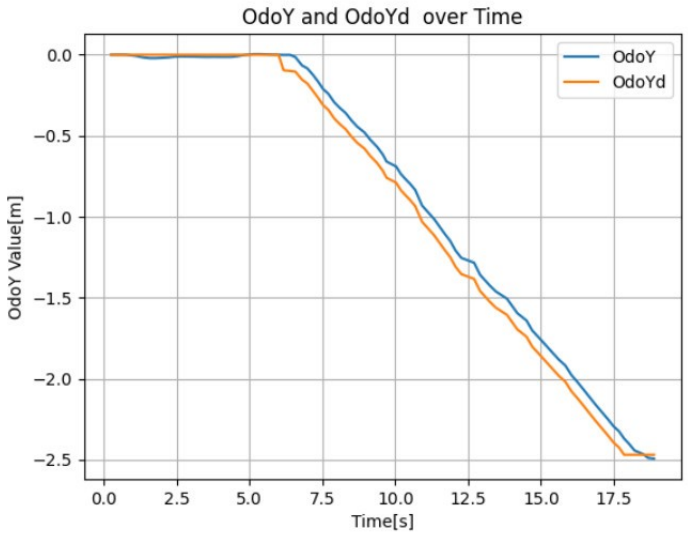


Fig. 82: *Y axis position response of vehicle over time*

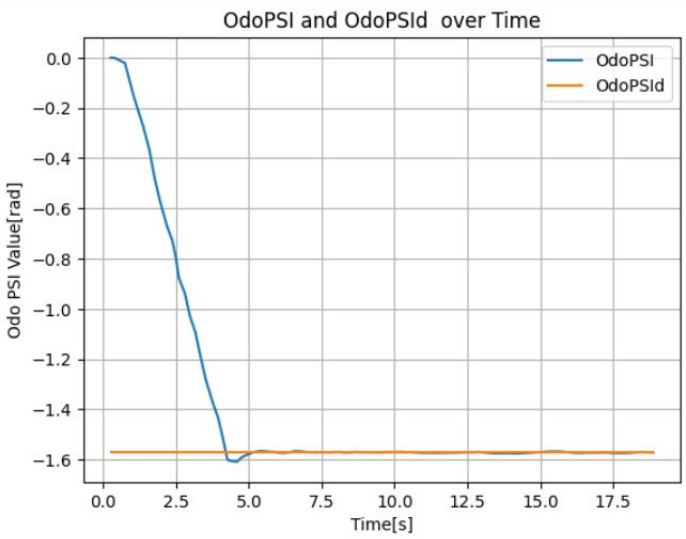


Fig. 83: *Angle response of vehicle over time*

6.3 Training the model

After labeling, the team proceeded to train the model. In terms of hardware, you used the NVIDIA GeForce RTX 3090 Ti 24GB graphics card for the model training

During the training process that spanned 300 epochs, Divide into steps with the magnitude of the loss, accuracy functions as follows:

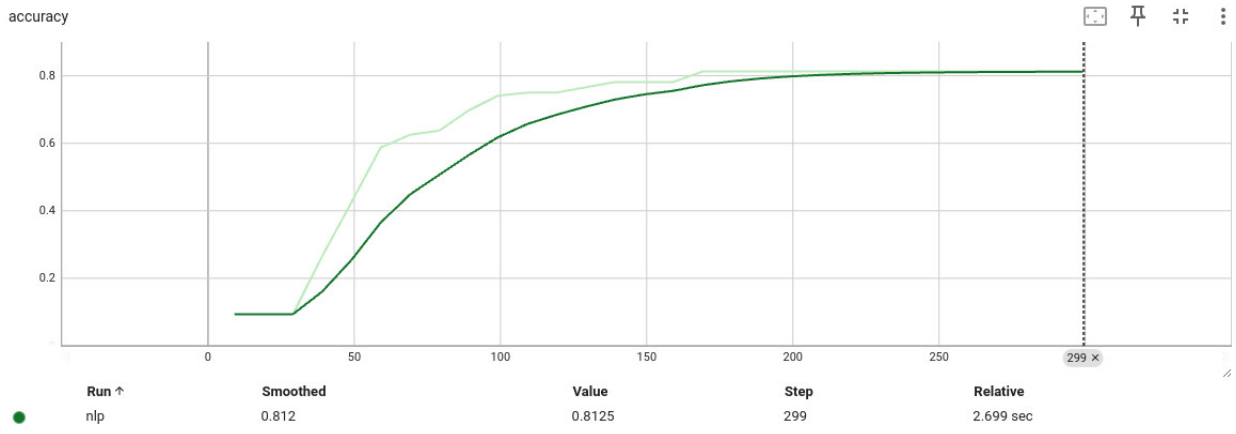


Fig. 84: *Accuracy function during training process*

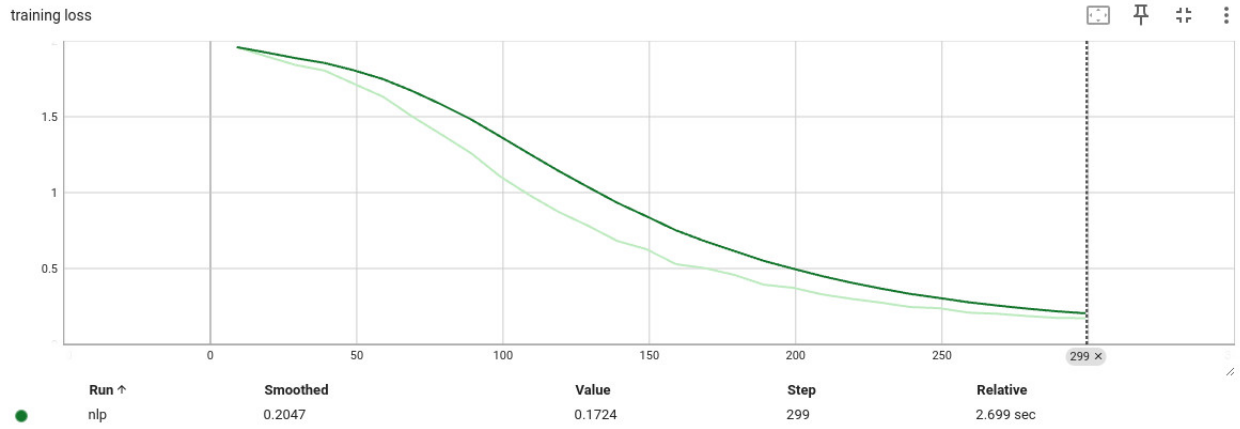
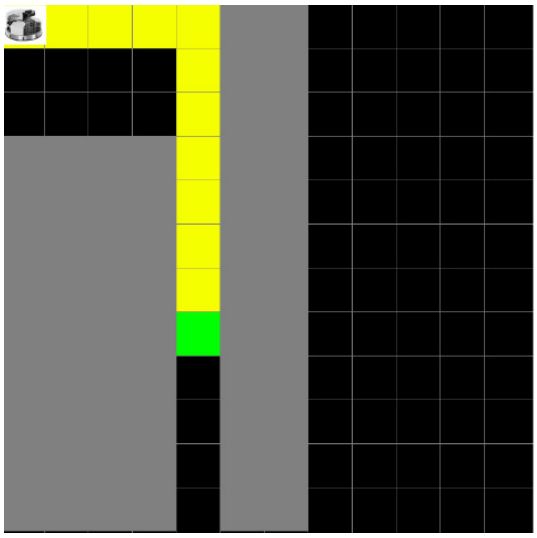


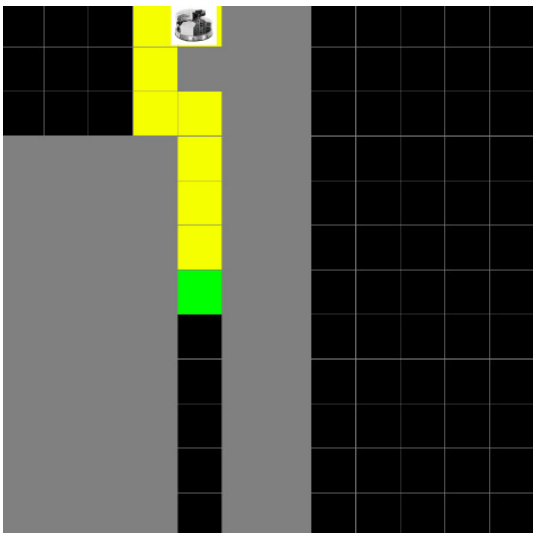
Fig. 85: *Loss function during training process*

6.4 Simulation A-Star

Our team simulates two cases where the path has no obstacles and is facing obstacles



(a) Without obstacles



(b) With obstacles

6.5 User interface designing

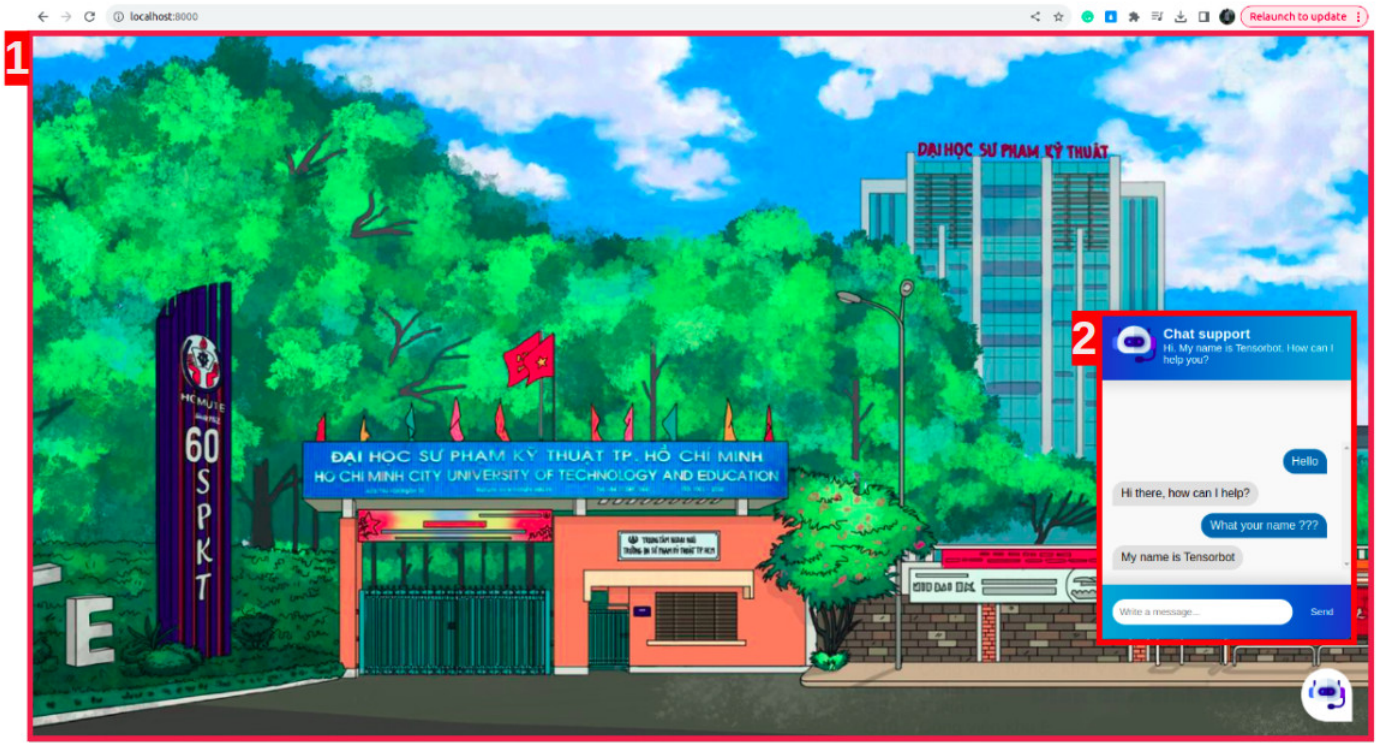


Fig. 86: *GUI (Graphical User Interface)*

- Region 1: To introduce the event of the school or department.
- Region 2: To interact with the chatbot to help us find information or lead to the room

7 CONCLUSION

On this topic, my group has completed the following tasks:

+ Completed the kinematic equation and verified it using matlab

+ Completed the kinematic equation and verified it using matlab

+ Solved the dynamic equation and found the relationship between the force acting on the wheels and the speed and acceleration of the system.

+ Understand and rewrite the pure pursuit algorithm, implemented in both simulation and reality Designed and applied point-to-point running algorithm in matlab simulation and reality

+ Understand and rewrite the shortest path algorithm

+ Complete the chatbot building model for users to interact

+ Completed construction of school introduction website

7.1 Summary:

7.2 Limitations and proposed solutions:

- Limitations :

+ Vehicle movement become unstable when network have problem

+ After a period of time ,there will be positional errors.

+ The vehicle may also produce vibrations or oscillations due to imperfect machining of the wheels.

+ Due to CPU hardware limitations, modern models such as Transformer, Bert or Attention cannot be used

+ Cannot operate in crowded places

+ Data is not yet large enough to respond flexibly

- Proposed solutions :

+ Change wifi router ,choose a network with better bandwidth,or choose a network with better connection quality,...

+ Optimize algorithm and Using Fusion Sensor: Combine data from multiple different sensors to estimate the position of the robot. Sensor fusion techniques can employ methods such as Kalman filtering or particle filtering to combine data from encoders, orientation sensors, lidar scanners, cameras, and any other available sensors to generate more accurate position estimates.

+ Optimize the parameters for the point control algorithm, or change the path tracking method such as MPC.

8 REFERENCES

[1] Joe Reavis, Robots aid homebound students, 2016,
https://sachsenews.com/2016/12/01/robots-aid-homebound-students

[2] Carly Banks, "Hello! Here is your delivery:" Starship robots roll onto NAU's campus, 2019,
https://news.nau.edu/starship-robots

[3] LG Business Solutions, CLOi GuideBot, Docent feature (optional),
https://www.lg.com/global/business/cloi/lg-cloi-guidebot

[4] Công ty TNHH Vạn Đạt, Sự khác biệt giữa truyền động dây curoa (belt drive) và truyền động dây xích (chain drive), Hình ảnh hệ truyền động dây curoa (belt drive), 2021,
https://vandat.com.vn/su-khac-biet-giua-truyen-dong-day-curoa-belt-drive-va-truyen-dong-day-xich-chain-drive

[5] Festo-Didactic InfoPortal, Robotino 4 Hardware > Control Unit,
https://ip.festo-didactic.com/InfoPortal/Robotino/Hardware/Controller/EN/index.html

[6] CadyTech, NXP, LPC2109FBD64,
https://www.cady-ic.com/product/LPC2109FBD64.html

[7] Microinfinity Co., R6093U SPECIFICATION Ver1.2, Figure 2: The CruizCore R6093U co-ordinate system,
https://ip.festo-didactic.com/InfoPortal/Robotino/document/gyro.pdf

[8] Pololu Robotics & Electronics, Sharp GP2Y0A41SK0F Analog Distance Sensor 4-30cm,
https://www.pololu.com/product/2464

[9] Festo-Didactic InfoPortal, Robotino 4 Hardware > Sensors,
https://ip.festo-didactic.com/InfoPortal/Robotino/Hardware/Sensors/EN/Bumper.html

[10] Direct Industry, Dunkermotoren GmbH Incremental rotary encoder RE 30 TI,
https://www.directindustry.com/prod/dunkermotoren-gmbh/product-14411-2010967.html

[11] Texas Instrument, LMD18200 3A, 55V H-Bridge, Figure 1 Functional Block Diagram of LMD18200,
https://www.ti.com/lit/ds/symlink/lmd18200.pdf

[12] Asokan Thondiyath, IIT Madras and Santhakumar Modhan, IIT palakkad NPTEL :: Mechanical Engineering - NOC : Wheeled Mobile Robots , 2020, https://archive.nptel.ac.in/courses/112/106/1

[13] Sarah Xiang from VRC team 97963A and VEXU team ILLINI, Basic Pure Pursuit,

https://wiki.purduesigbots.com/software/control-algorithms/basic-pure-pursuit

[14] M.Sc. Võ Lâm Chương ,Practice of Drive Servo Systems, Lecture note : MODEL 4. VELOCITY CONTROL FOR DC SERVO MOTORS, Ho Chi Minh City University of Technology and Education,2020

[15] Stuart Russell (Author), Peter Norvig (Author). Artificial Intelligence: A Modern Approach 3rd Edition. Publisher (December 1, 2009)

[16] Deepanshi, text Preprocessing in NLP, On April 26th, 2023
https://www.analyticsvidhya.com/blog/2021/06/text-preprocessing-in-nlp-with-python-codes/

[17] Deepanshi, A Brief Overview of Recurrent Neural Networks, On November 7th, 2023
https://www.analyticsvidhya.com/blog/2022/03/a-brief-overview-of-recurrent-neural-networks-rnn/

[18] Nguyen Thanh Tuan, Lesson 14 of series deep learning co ban, 2019
https://nttuan8.com/bai-14-long-short-term-memory-lstm/