الجامعة المصرية اليابانية للعلوم و التكنولوجيا
エジプト日本科学技術大学
EGYPT-JAPAN UNIVERSITY OF SCIENCE AND TECHNOLOGY

MTE 431
Mobile Robots

Group Course Project

Submitted by:

Mostafa Osama – ID:120200018

Yomna Mohamed – ID: 120200175

Omar Mohamed Fekry – ID:120200085

Abdallah Amr – ID:120200011

Tarek Shohdy – ID:120200030

Submitted to:

Dr. Haitham El-Hussieny

Contents:

I. Abstract

A Macnum wheel robot equipped with four omnidirectional wheels, designed to achieve seamless omnidirectional motion. The robotic system employs advanced control mechanisms, utilizing a PID controller for precise regulation of angular velocities. The key focus is on the calibration and testing process, emphasizing sensor calibration, PID tuning, and kinematic model validation. The exclusion of specific hardware components streamlines the calibration procedure, showcasing the significance of gyroscopes and accelerometers for accurate internal state estimation. The closed-loop control system demonstrates the robot's ability to move with some precise movement compared to the hardware used. The successful calibration and testing efforts lay the groundwork for the robot's reliable performance, offering promising prospects for further advancements in omnidirectional robotic control systems.

II. Kinematic Analysis

We used four Mecanum wheels configuration and all wheels are fixed, to model the kinematics of the robot, we need to study the constraints on the motion expressed by the wheel type.

Two constraints for each wheel type:

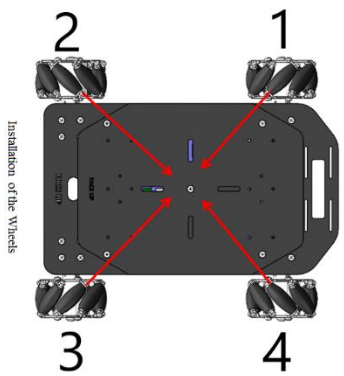
Rolling constraint: The wheel must roll when motion takes place in appropriate direction.

$$[\sin(\alpha+\beta+\gamma) \quad -\cos(\alpha+\beta+\gamma) \quad -l\cos(\beta+\gamma)] * \dot{\zeta}_R - r\dot{\emptyset}\cos(\gamma) = 0$$

No-slippage constraint: The wheel cannot slide laterally.

$$[\cos(\alpha+\beta+\gamma) \quad \sin(\alpha+\beta+\gamma) \quad -l\sin(\beta+\gamma)] * R(\theta)\dot{\zeta}_I - r\dot{\emptyset}\sin\gamma - r_{SW}\dot{\emptyset}_{SW} = 0$$

Where,



(l, α): defines the rotation axis position.

β: the wheel angle (fixed).

γ: rollers angle.

r_{SW} : roller radius.

$\dot{\emptyset}_{SW}$: roller speed.

So, we need to know α, β, γ for each wheel. And the dimensions of our kit are:

$$l = 18.1cm\,, r = 3cm$$

For First Wheel

$$\because \textit{The wheel is macnum}$$

$$\therefore \gamma_1 = -45°$$

$$\alpha_1 = \tan^{-1}\frac{149}{257} = 30°$$

$$\beta_1 = \tan^{-1}\frac{257}{149} = 60°$$

For Second Wheel

$$\because \textit{The wheel is macnum}$$

$$\therefore \gamma_2 = -135°$$

$$\alpha_2 = \tan^{-1}\frac{149}{-257} = 150°$$

$$\beta_2 = \tan^{-1}\frac{257}{149} = -60°$$

For Third Wheel

$$\because The\ wheel\ is\ macnum$$

$$\therefore \gamma_3 = -45°$$

$$\alpha_3 = 180 - \tan^{-1}\frac{149}{257} = 210°$$

$$\beta_3 = \tan^{-1}\frac{257}{149} = -120°$$

For Fourth Wheel

$$\because The\ wheel\ is\ macnum$$

$$\therefore \gamma_4 = -135°$$

$$\alpha_4 = 180 + \tan^{-1}\frac{149}{257} = -30°$$

$$\beta_4 = \tan^{-1}\frac{257}{149} = 120°$$

We will substitute α, β, γ for each wheel in Rolling Constraint equation:

For First Wheel:

$$[\sin(30 + 60 - 45) \quad -\cos(30 + 60 - 45) \quad 18.1 * \cos(60 - 45)] * \dot{\zeta}_R - r\dot{\emptyset}\cos(-45) = 0$$

For Second Wheel:

$$[\sin(150 - 60 - 135) \quad -\cos(150 - 60 - 135) \quad 18.1 * \cos(60 - 135)] * \dot{\zeta}_R - r\dot{\emptyset}\cos(-135) = 0$$

For Third Wheel:

$$[\sin(210 - 120 - 45) \quad -\cos(210 - 120 - 45) \quad 18.1 * \cos(-120 - 45)] * \dot{\zeta}_R - r\dot{\emptyset}\cos(-45) = 0$$

For Fourth Wheel:

$$[\sin(-30 + 120 - 135) \quad -\cos(-30 + 120 - 135) \quad 18.1 * \cos(120 - 135)] * \dot{\zeta}_R - r\dot{\emptyset}\cos(-135) = 0$$

$$\therefore \begin{bmatrix} 0.707 & -0.707 & -0.181 \\ -0.707 & -0.707 & 0.181 \\ 0.707 & -0.707 & 0.181 \\ -0.707 & -0.707 & -0.181 \end{bmatrix} * \begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta}_R \end{bmatrix} = \begin{bmatrix} 0.707r\dot{\emptyset}_1 \\ -0.707r\dot{\emptyset}_2 \\ 0.707r\dot{\emptyset}_3 \\ -0.707r\dot{\emptyset}_4 \end{bmatrix}$$

$$\begin{bmatrix} \dot{\emptyset}_1 \\ \dot{\emptyset}_2 \\ \dot{\emptyset}_3 \\ \dot{\emptyset}_4 \end{bmatrix} = \frac{1}{r} * \begin{bmatrix} 1 & -1 & -0.26 \\ 1 & 1 & -0.26 \\ 1 & -1 & 0.26 \\ 1 & 1 & 0.26 \end{bmatrix} * \begin{bmatrix} \dot{x}_R \\ \dot{y}_R \\ \dot{\theta}_R \end{bmatrix}$$

We made a python file that take $\dot{x}_R, \dot{y}_R, \dot{\theta}_R$ from joystick and give $\dot{\emptyset}_1, \dot{\emptyset}_2, \dot{\emptyset}_3, \dot{\emptyset}_4$ for each wheel.

III. Design choices and mechanical layout

i. Design choices

a. Mecanum Wheel Size and Type:
- Select wheels based on load capacity, durability, and intended use environment (indoor, outdoor, rough terrain, etc.).
- The size of the wheels affects the vehicle's clearance and maneuverability.

b. Chassis Configuration:
- Design a robust chassis that can support the weight of the wheels, the payload, and any additional components (like batteries, motors, etc.).
- The chassis should provide adequate mounting points for the wheels and motors.

c. Motor Selection:
- Choose motors with enough torque and speed capabilities to drive the wheels effectively.
- Each wheel should have an independent motor for precise control.

d. Battery and Power Management:
- Select a battery that matches the power requirements of the motors and any onboard electronics.
- Ensure proper power management systems are in place for efficiency and safety.

ii. Mechanical Layout

a. Wheel Placement:
- Arrange the wheels in a rectangular pattern, with each corner of the chassis equipped with one Mecanum wheel.

b. Chassis Design:
- The chassis should balance the weight distribution for stability.

c. Electronic Components Housing:
- Designate areas on the chassis for securely housing the control system, battery, and any additional electronics.

d. Wiring and Connectors:
- Plan the wiring layout to avoid interference with moving parts.

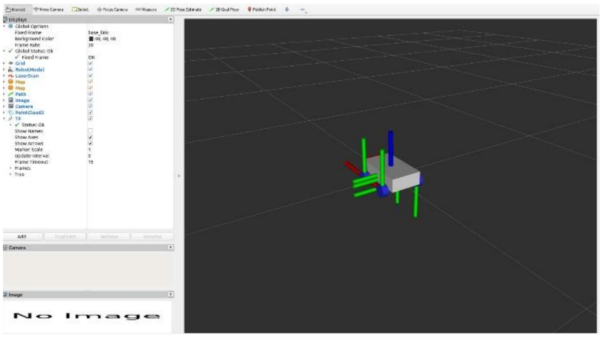
IV. Software implementation and challenges

i. Software Implementation:

a. Raspberry Pi Setup:
- The first step was to install the operating system that works with ROS1 noetic on raspberry pi, using a pre made image for raspberry pi.
- Network Configuration that ensures the Raspberry Pi is set up for network connectivity, essential for remote access and communication with other devices.
- Connecting with Arduino using rosserial to send the /wheel_vel topic to the Arduino and receive the 4 wheel speeds to move the robot.
- Connecting an Imu to raspberry pi and send the data to the laptob to do some control on the robots velocities.

b. Kinematic node:
- Develop a ROS node for calculating the 4 wheel velocities and send it to the Arduino using the calculations made in sec.1.

c. URDF Creation and simulation:
   - Making all the robots links and joints correctly, approximating the robot as a box with the wheels as cylinders to make the simulation easier to compute (the macnum wheel mesh will make the simulation slower)
   - Using a "planar motion" plugin for gazebo which make the robot move in omnidirectional motion.
   - Use a "libgazebo_ros_imu_sensor.so" plugin for gazebo to simulate an Inertial Measurement Unit (IMU) sensor in a Gazebo simulation environment.



d. Macnum bot launch file:
   - Launches all the previous node on the laptop with a bridge to the raspberry pi, but without using any control on the velocity (open loop control) and only controlling with the joystick.

e. PID Controller Implementation:
   - PID (Proportional-Integral-Derivative) controller for precise motor control that takes the setpoint from the cmd_vel from the joy node, the input as the imu measurements – angular velocity, yaw angle – and compute the angular velocity cmd of the robot to make sure the robot moves precisely in directions.

f. Controller Macnum bot launch file:
   - Launches the nodes in the macnum bot launch file adding the pid controller node with another node to set the Kp, Ki, Kd for the controller.

ii. Challenges:

a. Sensor calibration and Accuracy:
   - On of the most difficult challenges for any mobile robot the sensor calibration and the accuracy.
   - To overcome this, we used a pre-made and tested IMU package that calibrates itself and send a filtered data for the controller to assure precise movement.

b. Bridging Data
   - First, we used the normal ROS1 dynamic bridge to pass the msgs from the ros1 topics to ros2 topics and vice versa, but this wasn’t efficient as the bridge was too slow and laggy and the data was sent in a much less frequency than the source.
   - The solution for this was to implement our own customized bridge using local port between 2 python nodes, one over the ros1 node and the other in ros2 node.

c. PID Tuning
   - With the help of the dynamic reconfigure, we tried to tune the Kp, Ki, Kd for the controller while the robot is moving and we tried to reach optimal values for the robots movement.
   - The environment changes will be some difficult to overcome as not all the floors will give the same response with the wheels.

V. Testing and Calibrating:

i. Sensor Calibration:

- **Gyroscopes and Accelerometers:** Calibrate IMU to enhance the accuracy of the robot's internal state estimation.

- **Results:** Imu data sent from the node are reliable and precise enough at least for the angular velocity and yaw angle.

ii. Open-Loop Control Testing:

- **Individual Wheel Testing:** Execute open-loop control tests for each wheel separately to observe their responses to specified angular velocity commands. Verify that each wheel can achieve the desired velocities.
- **Overall System Response:** Test the robot's response to open-loop control commands for various motion patterns (e.g., straight-line motion, turning) to identify any discrepancies in individual wheel behavior.
- **Inverse Kinematics:** Validate the inverse kinematic model by commanding the robot to move in any direction and verifying that the calculated wheel velocities result in the desired configuration.
- **Results:** the unidentical motors lead to a less expected behavior from the robot, as it will move in the x direction correctly, but it will deviate and turn and the speed in the x direction will not be the same as given.

iii. Closed-Loop Control Testing:

- **Proportional (P), Integral (I), and Derivative (D) Components:** Fine-tune the PID controller parameters for optimal angular velocity control. Adjust the P, I, and D gains to achieve a balance between system responsiveness, stability, and elimination of steady-state error.
- **Stability Testing:** Evaluate the stability of the PID-controlled system under different operating conditions, ensuring it does not exhibit overshooting or oscillations.
- **Angular Velocity Tracking:** Implement closed-loop control with PID to regulate the angular velocity of the robot. Observe how well the robot follows the desired angular velocity commands.
- **Results:** The robot is quiet stable, but with the changed environment and all the approximations in the robot model, it’s difficult to achieve some perfect response from the system.

VI. Conclusion

In conclusion, the calibration and testing of the Macnum wheel robot, emphasizing PID-controlled angular velocity and kinematic model validation, have resulted in a reliable system. By focusing on sensor calibration, PID controller tuning, and comprehensive testing, we can achieve more precise angular velocity regulation and validated the accuracy of the robot's kinematic models.

The validation of inverse kinematics confirmed the theoretical representations closely align with observed behavior, enhancing our understanding of the robot's motion. Meticulous documentation of calibration settings and test results provides a valuable resource for troubleshooting and future development.

In summary, the Macnum wheel robot has undergone a successful calibration, resulting in a well-functioning robotic system. The combination of accurate sensor calibration, tuned PID control, and validated kinematic models establishes a solid foundation for its performance in real-world applications, paving the way for further advancements in robotic control systems.