

**CENTRO UNIVERSITÁRIO FEI**

**MURILO DARCE BORGES SILVA**

**RODRIGO SIMÕES RUY**

**IDENTIFICAÇÃO DE DIFERENÇAS DE DESEMPENHO ENTRE ROBÔS REAIS,  
CONTEINERIZADOS E SIMULADOS, UTILIZANDO GAZEBO **SIMULATOR**, ROS  
2 E DOCKER.**

São Bernardo do Campo

2025

MURILO DARCE BORGES SILVA  
RODRIGO SIMÕES RUY

**IDENTIFICAÇÃO DE DIFERENÇAS DE DESEMPENHO ENTRE ROBÔS REAIS,  
CONTEINERIZADOS E SIMULADOS, UTILIZANDO GAZEBO SIMULATOR, ROS  
2 E DOCKER.**

Trabalho de Conclusão de Curso apresentado ao Centro Universitário FEI, como parte dos requisitos necessários para obtenção do título de Bacharel em Ciência da Computação. Orientado pelo Prof. Dr. Leonardo Anjoletto Ferreira.

São Bernardo do Campo  
2025

MURILO DARCE BORGES SILVA  
RODRIGO SIMÕES RUY

**IDENTIFICAÇÃO DE DIFERENÇAS DE DESEMPENHO ENTRE ROBÔS REAIS,  
CONTEINERIZADOS E SIMULADOS, UTILIZANDO GAZEBO SIMULATOR, ROS  
2 E DOCKER.**

Trabalho de Conclusão de Curso apresentado ao  
Centro Universitário FEI, como parte dos requisitos  
necessários para obtenção do título de Bacharel em  
Ciência da Computação.

Comissão julgadora

Prof. Dr. Leonardo Anjoletto Ferreira

Prof. Dr. Plinio Thomaz Aquino Junior

Profa. Dra. Fagner de Assis Moura Pimentel

São Bernardo do Campo

06/06/2025

## **RESUMO**

Containerização é uma ferramenta muito útil quando se lida com projetos que precisam de diferentes dependências ou programas que podem ter conflitos entre si, que precisam de uma grande quantidade de configuração inicial, ou que precisam de portabilidade. Isso a torna perfeita para projetos de robótica, mas os impactos do seu uso e suas peculiaridades em situações reais ainda não estão documentadas, o que é justamente o que este projeto propõe fazer. Haverão 2 partes para este projeto, a primeira parte será uma avaliação do desempenho em uma simulação utilizando Gazebo, e a segunda parte será a avaliação do desempenho de um turtlebot real.

Palavras-chave: Robótica, ROS, Docker, Containerização

## **ABSTRACT**

Containerization is a very useful tool when dealing with projects that require different dependencies or programs that may conflict with each other, require a lot of initial configuring, or demand portability. This made it perfect for robotics projects, but the impacts of its usage and quirks in real scenarios are still undocumented, which is what this project aims to do. The project will have 2 parts, the first will be an assessment of the performance of a simulation using Gazebo, and the second part will be an assessment of the performance of a real turtlebot.

**Keywords:** Robotics, ROS, Docker, Containerization

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO</b>	<b>5</b>
1.1	OBJETIVO	5
<b>2</b>	<b>CONCEITOS FUNDAMENTAIS</b>	<b>6</b>
<b>2.0.1</b>	<b>FERRAMENTAS</b>	<b>6</b>
<b>3</b>	<b>TRABALHOS RELACIONADOS</b>	<b>9</b>
<b>4</b>	<b>METODOLOGIA</b>	<b>10</b>
<b>5</b>	<b>EXPERIMENTOS E RESULTADOS</b>	<b>11</b>
	REFERÊNCIAS	17

# 1 INTRODUÇÃO

A containerização é uma ferramenta poderosa no campo de desenvolvimento e implementação, disponibilizando certa camada de isolamento entre componentes de um projeto, assegurando que estes não irão conflitar, seja por funções internas ou dependências de versões diferentes sendo utilizadas. No campo da robótica, containerização é vista como uma técnica para facilitar o desenvolvimento, portabilidade e consistência em projetos de robótica, mas não foram feitas pesquisas detalhando a integração destes projetos com Docker e seus efeitos no desempenho de um robô físico. A proposta do projeto é justamente esta: integrar ROS 2 e Docker, explicando os passos utilizados e comparando o desempenho com e sem containerização, sendo dividido em 2 partes: em um simulador Gazebo, e em um robô Turtlebot real.

## 1.1 OBJETIVO

Documentar e avaliar o desempenho do robô simulado e do robô real quando comparados à sua implementação com e sem containerização.

## 2 CONCEITOS FUNDAMENTAIS

Para o entendimento de alguns softwares e o porquê de estarem sendo utilizados, existem alguns conceitos que devem ser compreendidos para o total entendimento do projeto. Estes conceitos são:

**Containerização:** Uma forma de virtualização feita para ser mais rápida e flexível que emulação, mas ainda ter sua consistência e isolamento do Sistema Operacional principal (WEN et al., 2023).

**Meta-sistema operacional:** É um sistema operacional que é desenvolvido e executado em outro sistema operacional, permitindo que diferentes processos possam se comunicar durante a execução. Este conceito é aplicado para o ROS.

**TurtleBot:** É um kit de robô pessoal que possui um preço acessível para o público. O robô é de código aberto, permitindo que seus usuários possam desenvolver as aplicações que desejarem utilizar no mesmo. Foi desenvolvido para educação e pesquisa do ROS. O kit possui sensores de distância 2D/3D, um computador embarcado, o robô e mais acessórios para o uso e utilização do robô(FOUNDATION, 2025).

### 2.0.1 FERRAMENTAS

Para o desenvolvimento do projeto, serão utilizados softwares para que os testes possam ser realizados e analisados. Estes testes serão:

**Docker:** É uma plataforma utilizada para desenvolvimento, envio e funcionamento de aplicações de maneira separada da infraestrutura por conta da containerização, por conta deste fator o usuário consegue gerir as aplicações da mesma maneira que gera sua infraestrutura, outro fator importante é que o Docker permite que as aplicações desenvolvidas sejam testadas e executadas com menos atraso do que a maneira convencional. Contêineres são bons para fluxos de integrações e entregas de trabalho contínuas.(INC., 2025)

**ROS2(Robot Operating System 2):** É um meta-sistema operacional de código aberto utilizado para auxiliar a desenvolver aplicações para robôs, o mesmo possui serviços que outros sistemas operacionais normalmente possuem, mas com o foco maior para a área da robótica, facilitando comunicação entre processos, funções que se comunicam com as demais e entre muitos outros. Para o desenvolvimento do projeto, será utilizado o ROS 2 que mantém o con-



ceito modular e distribuído, mas possui melhorias e mais funcionalidades que o ROS original (Observar figura 1) (FOUNDATION, 2018)

**Gazebo Simulator:** É um software usado para desenvolver simulações, possui diversos projetos de código aberto para que os interessados possam utilizar e desenvolver suas próprias simulações. Neste software estão presentes também diversos modelos, tanto como objetos como também robôs. (Observar figura 2) (FOUNDATION, 2022)

**TurtleBot3 Burgers:** É um robô customizável e de preço acessível ao público baseado no modelo ROS para ser utilizado na educação, pesquisas e entretenimento pessoal, é um robô que possui o intuito de ser barato, por conta disto, o mesmo não possui uma grande funcionalidade ou qualidade, mas o mesmo compensa na relação da quantidade de aplicações que o mesmo consegue realizar. (Observar figura 3) (ROBOTIS, 2025)

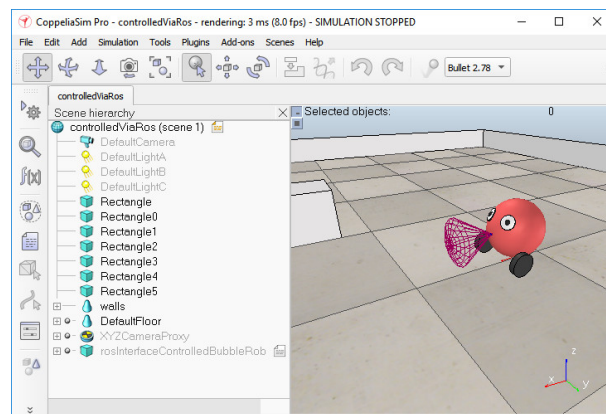


Figura 1 – Exemplo do Robot Operating System (COPPELIASIM, 2020)

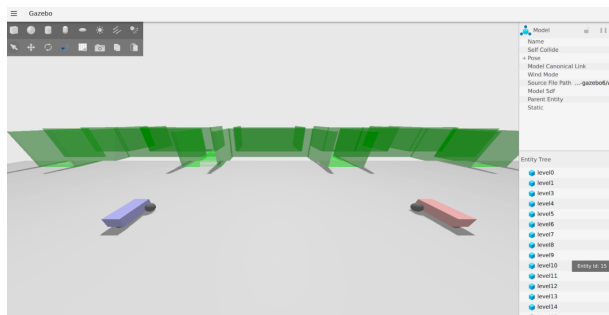


Figura 2 – Exemplo utilizando o Gazebo Simulator (FOUNDATION, 2022)

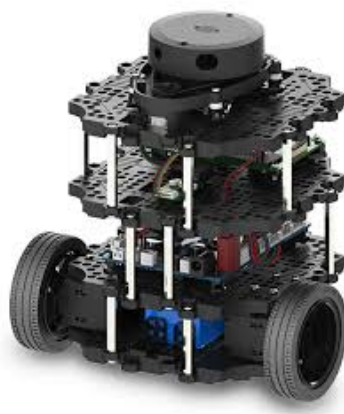



Figura 3 – TurtleBot3 Burger (FOUNDATION, 2025)

### 3 TRABALHOS RELACIONADOS

Para o desenvolvimento do projeto, foram relacionados alguns artigos que auxiliam no desenvolvimento do mesmo. O artigo "Bare-Metal vs. Hypervisors and Containers: Performance Evaluation of Virtualization Technologies for Software-Defined Vehicles"(WEN et al., 2023) auxilia com relação ao entendimento da containerização em sistemas embarcados e também com relação ao seu desempenho. No artigo, **é detalhada a utilização** de diferentes formas de containerização e seu efeito no desempenho em diferentes tipos de hardware. ~~O~~ artigo "Docker Performance Evaluation across Operating Systems"(SOBIERAJ; KOTYŃSKI, 2024) auxilia no entendimento dos **conceitos de avaliação do Docker** com relação a outros sistemas operacionais, para verificar a diferença entre os sistemas operacionais o mesmo utilizou os recém instalados Sistemas Operacionais que eram MacOS ventura, Ubuntu 22.04 e Windows 10 rodando em um MacBook Pro 13, os testes consistiam em "estressar"o Docker com relação à CPU, Rede e na resiliência do mesmo. ~~este~~ artigo auxilia no **entendimento com relação aos tipos de testes que podem ser realizados** para analisar o desempenho dos robôs nas simulações e nos testes físicos. ~~Para~~ a parte física, o artigo "Design and Ground Performance Evaluation of a Multi-Joint Wheel-Track Composite Mobile Robot for Enhanced Terrain Adaptability"(GAO et al., 2023) tem o intuito de propor um robô multi-junta que consiga circular em vários tipos de terrenos, o artigo se diferencia por ser um robô com pernas  mas demonstra maneiras de se analisar o desempenho do robô auxiliando com relação à análise e entendimento de desempenho de um robô físico e sobre as suas causas.

## 4 METODOLOGIA

A metodologia é dividida em 2 partes. Na primeira parte, será utilizada uma simulação no gazebo simulator. A simulação possuirá um ambiente para testar o desempenho de um robô turtlebot. Ocorrerão dois testes, no primeiro teste o robô utilizará apenas ROS 2 enquanto no segundo teste o robô utilizará ROS 2 e Docker. Após a **análise do desempenho** nos testes, os dados serão armazenados, analisados e finalmente comparados. Na segunda parte, ao invés de uma simulação, será utilizado um robô turtlebot real. Os testes são semelhantes aos aplicados na primeira parte, onde o primeiro teste será realizado apenas com ROS 2 enquanto o segundo será realizado com ROS 2 e Docker juntos, para que assim os dados de desempenho possam ser coletados, analisados e então comparados. As arenas que serão utilizadas na simulação serão baseadas em possíveis arenas que **serão montadas** na FEI na sala k404. A mesma possui uma arena que pode ser montada e um robô TurtleBot 3 Burger para a realização dos testes, assim será desenvolvida uma simulação com uma arena baseada na arena existente na instituição. A arena possui um caminho, mas **o mesmo pode ser alterado** com algumas placas que funcionam como paredes, essas paredes permitem a criação de um caminho diferente para o robô podendo ser feito um labirinto, assim seria desenvolvida uma simulação com a mesma ideia para que se pudesse obter dados com uma relação maior. Com relação à parte física, ~~em algum horário disponível e com as devidas autorizações~~, seriam utilizados tanto o robô quanto a arena física para que pudessem ser executados os testes necessários para se adquirir os dados para então analisá-los e compararmos.

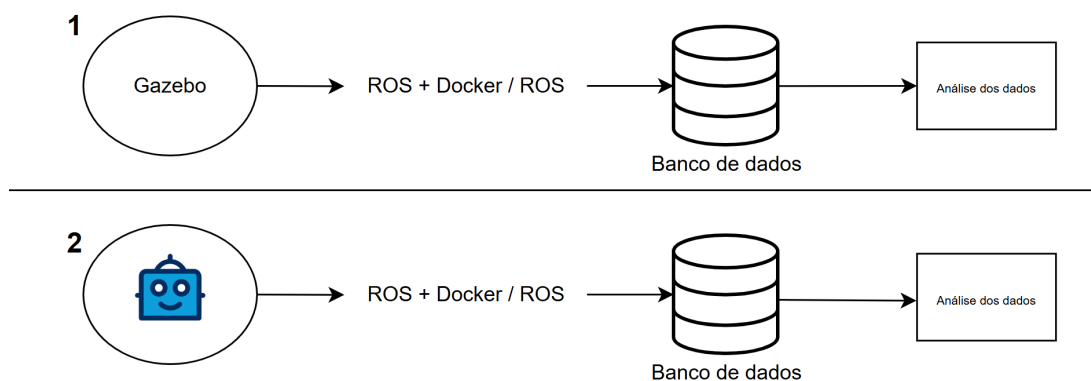


Figura 4 – Fluxograma do projeto



## 5 EXPERIMENTOS E RESULTADOS

Durante o TCC1, já foram feitos testes para implementar a parte simulada proposta, sendo utilizado o manual (ROBOTIS, 2025) e packages (ROBOTIS-GIT, 2025) disponíveis pelo grupo ROBOTIS, tornando o desenvolvimento desta parte rápido. A utilização e desenvolvimento dos projetos ROS2 dentro do Docker foi facilitada pela utilização de workflows no GitHub, onde as imagens de teste foram automaticamente construídas e publicadas como packages no repositório, o que reduziu o tempo do processo de construir as imagens localmente, que demorava de 20+ minutos para no máximo 5 minutos, além de também as disponibilizar para outros usarem.

Houve certas dificuldades para integrar o ROS2 dentro do contêiner Docker com o ROS2 nativo, que lidaria com a simulação Gazebo. Foi descoberto que o middleware utilizado por padrão pelo ROS2 Humble, FastDDS, não interage de forma consistente com Docker. Ele era capaz de compartilhar os tópicos entre os ambientes, mas causava falha na publicação e recebimento de mensagens, não mostrando nenhuma.

Para solucionar isso, o FastDDS foi substituído por outro middleware disponível para ROS2 Humble, CycloneDDS, o qual foi utilizado especificamente no contêiner Docker.

```

rodriago@BattleStation25: ~
root@BattleStation25: /home/rodriago/...
rodriago@BattleStation25: ~
rodriago@BattleStation25: $ echo $RMW_IMPLEMENTATION
rodriago@BattleStation25: $ ros2 topic list
/parameter_events
/rosout
rodriago@BattleStation25: $ ros2 run demo_nodes_cpp talker
[INFO] [1748231727.656969225] [talker]: Publishing: 'Hello World: 1'
[INFO] [1748231728.656933660] [talker]: Publishing: 'Hello World: 2'
[INFO] [1748231729.656980784] [talker]: Publishing: 'Hello World: 3'
[INFO] [1748231730.657285389] [talker]: Publishing: 'Hello World: 4'
[INFO] [1748231731.657301086] [talker]: Publishing: 'Hello World: 5'
[INFO] [1748231732.657292011] [talker]: Publishing: 'Hello World: 6'
[INFO] [1748231733.657285603] [talker]: Publishing: 'Hello World: 7'
[INFO] [1748231734.657286758] [talker]: Publishing: 'Hello World: 8'
[INFO] [1748231735.657267963] [talker]: Publishing: 'Hello World: 9'
[INFO] [1748231736.657290746] [talker]: Publishing: 'Hello World: 10'
[INFO] [1748231737.656933981] [talker]: Publishing: 'Hello World: 11'
[INFO] [1748231738.656962986] [talker]: Publishing: 'Hello World: 12'
[INFO] [1748231739.656961720] [talker]: Publishing: 'Hello World: 13'
[INFO] [1748231740.657492615] [talker]: Publishing: 'Hello World: 14'
[INFO] [1748231741.657030920] [talker]: Publishing: 'Hello World: 15'
[INFO] [1748231742.656963033] [talker]: Publishing: 'Hello World: 16'
[INFO] [1748231743.657117598] [talker]: Publishing: 'Hello World: 17'
[INFO] [1748231744.657020486] [talker]: Publishing: 'Hello World: 18'

rodriago@BattleStation25: ~
root@BattleStation25: /dockerteste
root@BattleStation25: $ sudo su
[sudo] password for rodriago:
root@BattleStation25: /home/rodriago# docker exec -it tcc-ros2-1 bash
root@BattleStation25: /dockerteste# ls
src
root@BattleStation25: /dockerteste# echo $RMW_IMPLEMENTATION
rodriago@BattleStation25: /dockerteste# ros2 topic list
/parameter_events
/rosout
root@BattleStation25: /dockerteste# ros2 topic list
/chatter
/parameter_events
/rosout
root@BattleStation25: /dockerteste# ros2 topic echo /chatter

```

Figura 5 – Falha utilizando RMW padrão (FastDDS)

```

root@BattleStation25: ~
[INFO] [1748231764.656956365] [talker]: Publishing: 'Hello World: 38'
[INFO] [1748231765.656955442] [talker]: Publishing: 'Hello World: 39'
[INFO] [1748231766.656956784] [talker]: Publishing: 'Hello World: 40'
[INFO] [1748231767.656939348] [talker]: Publishing: 'Hello World: 41'
[INFO] [1748231768.656924168] [talker]: Publishing: 'Hello World: 42'
[INFO] [1748231769.656928444] [talker]: Publishing: 'Hello World: 43'
[INFO] [1748231770.656928206] [talker]: Publishing: 'Hello World: 44'
[INFO] [1748231771.657078324] [talker]: Publishing: 'Hello World: 45'
[INFO] [1748231772.657159519] [talker]: Publishing: 'Hello World: 46'
[INFO] [1748231773.657244936] [talker]: Publishing: 'Hello World: 47'
[INFO] [1748231774.657329944] [talker]: Publishing: 'Hello World: 48'
^C[INFO] [1748231775.326495205] [rclcpp]: signal_handler(signum=2)
root@BattleStation25: ~$ ros2 topic list
/parameter_events
/rosout
root@BattleStation25: ~$ ros2 run demo_nodes_cpp talker
[INFO] [1748231966.319745640] [talker]: Publishing: 'Hello World: 1'
[INFO] [1748231967.319316139] [talker]: Publishing: 'Hello World: 2'
[INFO] [1748231968.319311374] [talker]: Publishing: 'Hello World: 3'
[INFO] [1748231969.319424812] [talker]: Publishing: 'Hello World: 4'
[INFO] [1748231970.319747799] [talker]: Publishing: 'Hello World: 5'
[INFO] [1748231971.319770725] [talker]: Publishing: 'Hello World: 6'
[INFO] [1748231972.319873304] [talker]: Publishing: 'Hello World: 7'
[INFO] [1748231973.319813529] [talker]: Publishing: 'Hello World: 8'

root@BattleStation25: /dockerteste
/chatter
/parameter_events
/rosout
root@BattleStation25: /dockerteste# ros2 topic echo /chatter
^Croot@BattleStation25: /dockerteste# ros2 topic list
/parameter_events
/rosout
root@BattleStation25: /dockerteste# export RMW_IMPLEMENTATION=rmw_cyclonedds_cpp
root@BattleStation25: /dockerteste# ros2 topic list
/chatter
/parameter_events
/rosout
root@BattleStation25: /dockerteste# ros2 topic echo /chatter
data: 'Hello World: 18'
---
data: 'Hello World: 19'
---
data: 'Hello World: 20'
---
data: 'Hello World: 21'
---
data: 'Hello World: 22'
---
data: 'Hello World: 23'
---
data: 'Hello World: 24'

```

Figura 6 – Sucesso utilizando RMW CycloneDDS

```

Open  compose.yml  Save
~/TCC

1 services:
2   ros2:
3     image: ghcr.io/joca2511/tcc_docker:main
4     network_mode: host
5     privileged: true
6     environment:
7       - ROS_DOMAIN_ID=10
8       - TURTLEBOT3_MODEL=burger
9       - RMW_IMPLEMENTATION=rmw_cyclonedds_cpp
10    stdin_open: true
11    tty: true

```

Figura 7 – Compose proposto para testes

```

root@BattleStation25: ~
root@BattleStation25: /home/rodrigo/...
root@BattleStation25: ~
rodrigo@BattleStation25:~$ export TURTLEBOT3_MODEL=burger
rodrigo@BattleStation25:~$ ros2 launch turtlebot3_gazebo turtlebot3_house.launch
.py

```

Figura 8 – Simulação Gazebo é lançada pelo host

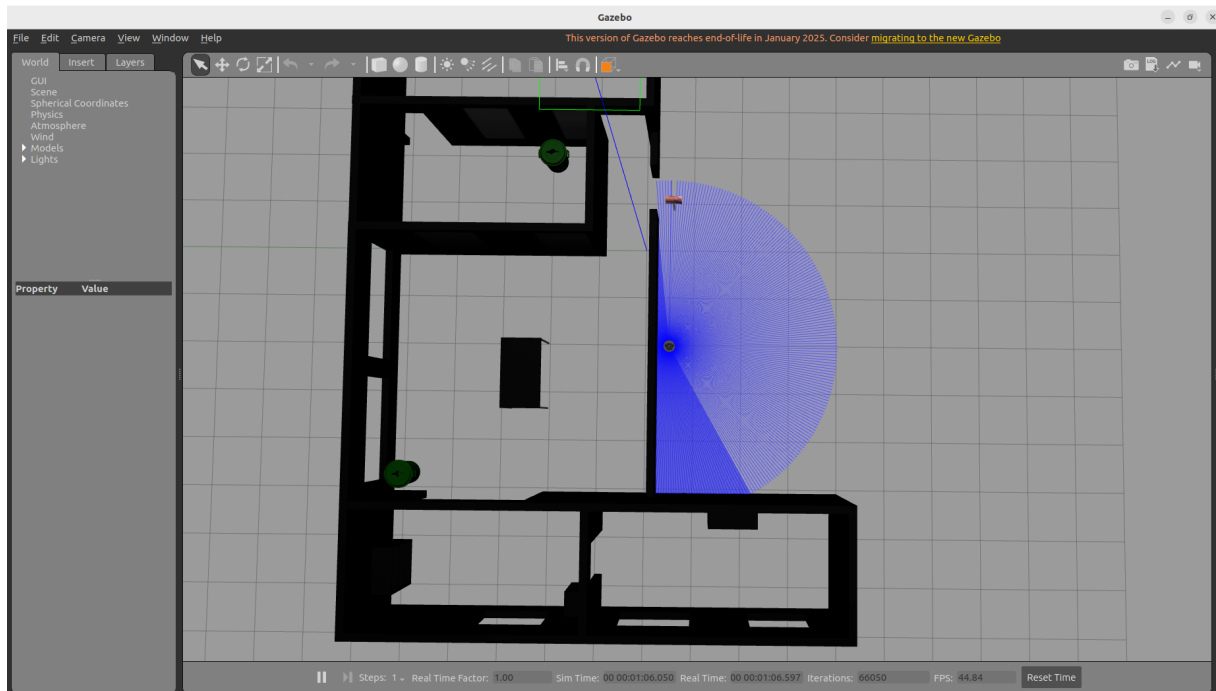


Figura 9 – Aparência inicial da simulação Gazebo

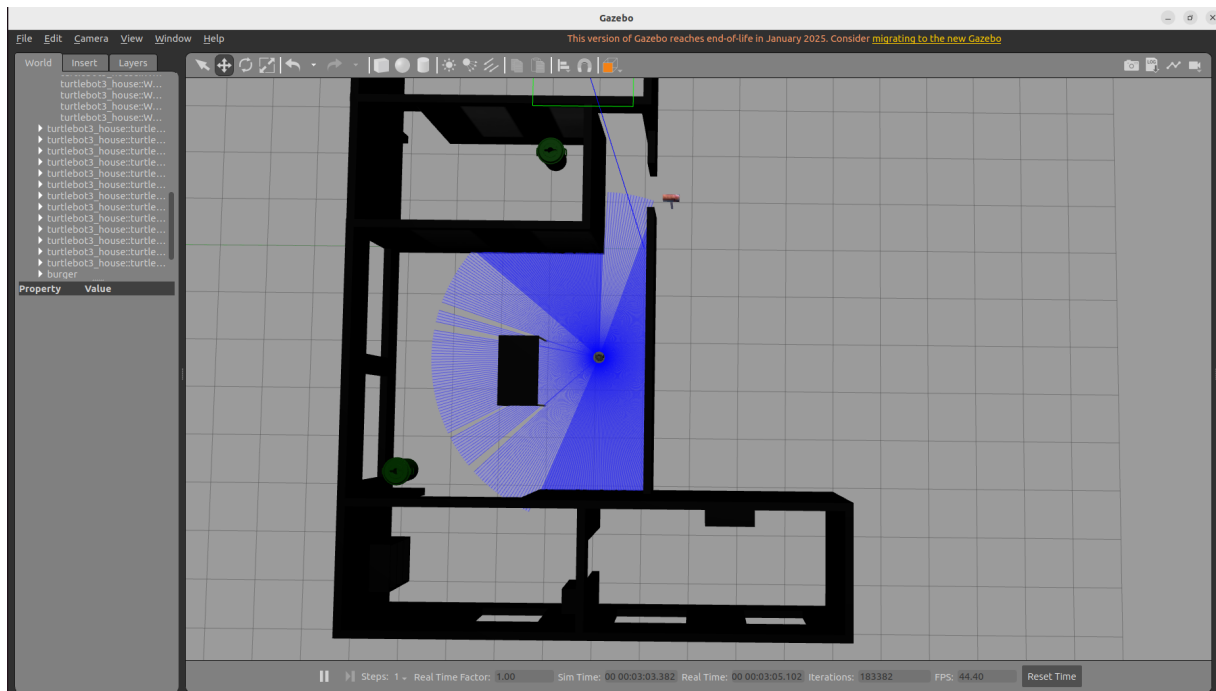
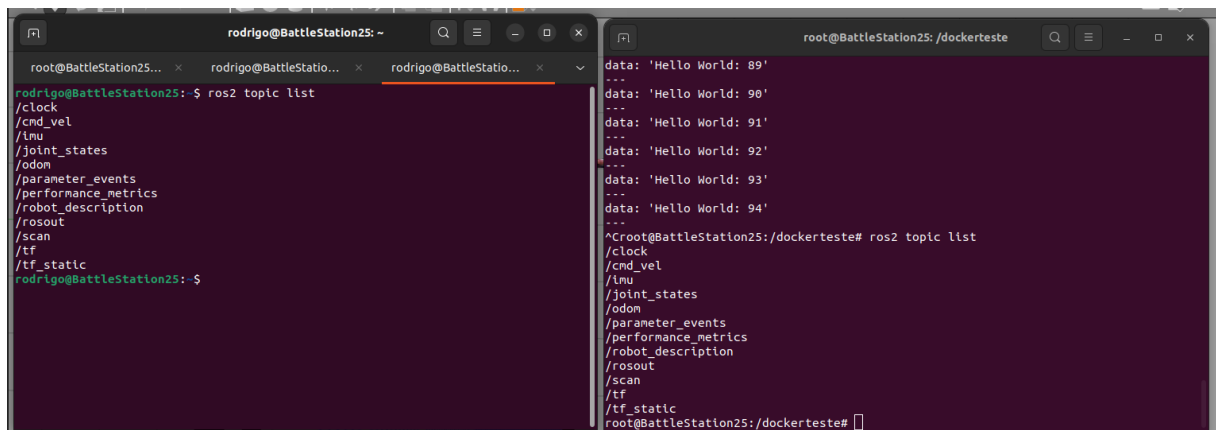


Figura 10 – Turtlebot3 Burger é movido para dentro da casa



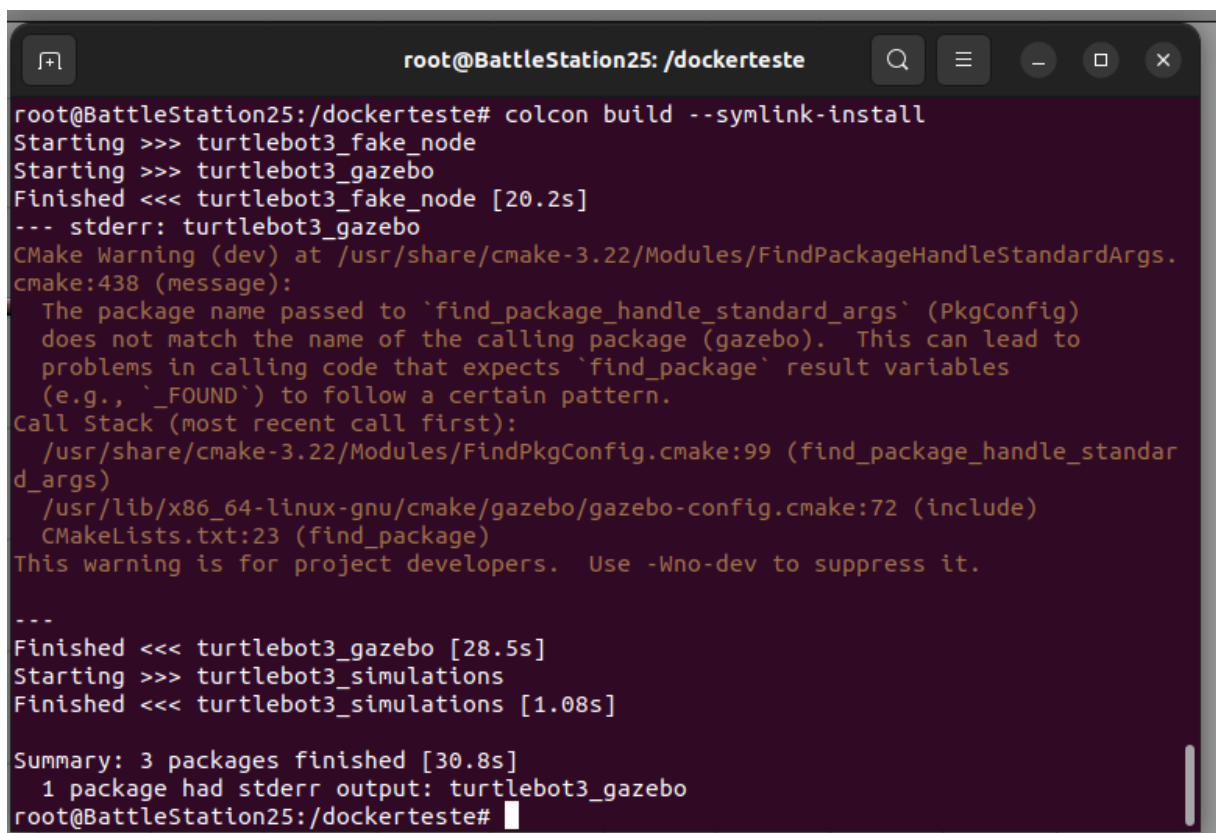
```

root@BattleStation25:~$ ros2 topic list
/clock
/cmd_vel
/lm
/joint_states
/odometer
/parameter_events
/performance_metrics
/robot_description
/rosout
/scan
/tf
/tf_static
root@BattleStation25:~$

root@BattleStation25:/dockerteste# ros2 topic list
data: 'Hello World: 89'
---
data: 'Hello World: 90'
---
data: 'Hello World: 91'
---
data: 'Hello World: 92'
---
data: 'Hello World: 93'
---
data: 'Hello World: 94'
---
^Croot@BattleStation25:/dockerteste#

```

Figura 11 – ROS2 Humble dentro do Docker adquire os novos tópicos



```

root@BattleStation25:/dockerteste# colcon build --symlink-install
Starting >>> turtlebot3_fake_node
Starting >>> turtlebot3_gazebo
Finished <<< turtlebot3_fake_node [20.2s]
--- stderr: turtlebot3_gazebo
CMake Warning (dev) at /usr/share/cmake-3.22/Modules/FindPackageHandleStandardArgs.
cmake:438 (message):
  The package name passed to `find_package_handle_standard_args` (PkgConfig)
  does not match the name of the calling package (gazebo). This can lead to
  problems in calling code that expects `find_package` result variables
  (e.g., `_FOUND`) to follow a certain pattern.
Call Stack (most recent call first):
  /usr/share/cmake-3.22/Modules/FindPkgConfig.cmake:99 (find_package_handle_standard
d_args)
  /usr/lib/x86_64-linux-gnu/cmake/gazebo/gazebo-config.cmake:72 (include)
  CMakeLists.txt:23 (find_package)
This warning is for project developers. Use -Wno-dev to suppress it.

---
Finished <<< turtlebot3_gazebo [28.5s]
Starting >>> turtlebot3_simulations
Finished <<< turtlebot3_simulations [1.08s]

Summary: 3 packages finished [30.8s]
1 package had stderr output: turtlebot3_gazebo
root@BattleStation25:/dockerteste#

```

Figura 12 – ROS2 Humble dentro do Docker builda projetos



```

root@BattleStation25: /dockerteste

--- stderr: turtlebot3_gazebo
CMake Warning (dev) at /usr/share/cmake-3.22/Modules/FindPackageHandleStandardArgs.
cmake:438 (message):
  The package name passed to 'find_package_handle_standard_args' (PkgConfig)
  does not match the name of the calling package (gazebo). This can lead to
  problems in calling code that expects 'find_package' result variables
  (e.g., '_FOUND') to follow a certain pattern.
Call Stack (most recent call first):
  /usr/share/cmake-3.22/Modules/FindPkgConfig.cmake:99 (find_package_handle_standar
d_args)
  /usr/lib/x86_64-linux-gnu/cmake/gazebo/gazebo-config.cmake:72 (include)
  CMakeLists.txt:23 (find_package)
This warning is for project developers. Use -Wno-dev to suppress it.

---
Finished <<< turtlebot3_gazebo [28.5s]
Starting >>> turtlebot3_simulations
Finished <<< turtlebot3_simulations [1.08s]

Summary: 3 packages finished [30.8s]
1 package had stderr output: turtlebot3_gazebo
root@BattleStation25:/dockerteste# source install/setup.bash
root@BattleStation25:/dockerteste# ros2 run turtlebot3_gazebo turtlebot3_drive
[INFO] [1748232703.578375574] [turtlebot3_drive_node]: Turtlebot3 simulation node h
as been initialised

```

Figura 13 – ROS2 Humble dentro do Docker roda projetos para movimentação do robô dentro da simulação Gazebo presente no host

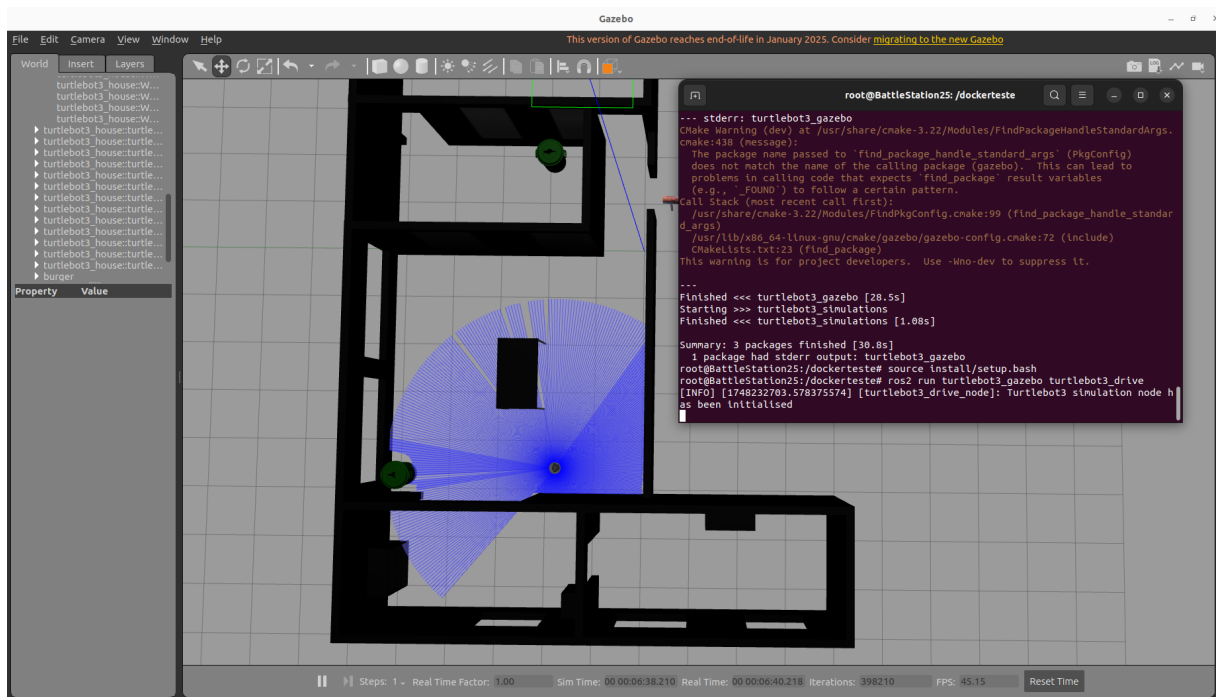


Figura 14 – Turtlebot3 Burger é movimentado pelo ROS2 Humble presente dentro do container Docker

Figura 15 – Turtlebot3 Burger é movimentado pelo ROS2 Humble presente dentro do container Docker





## REFERÊNCIAS

COPPELIASIM. **ROS2 tutorial**. 2020. Accessed: 2025-05-25. Disponível em: <<https://manual.coppeliarobotics.com/en/ros2Tutorial.htm>>.

FOUNDATION, I. O. S. R. **ROS**. 2018. Accessed: 2025-05-24. Disponível em: <<https://wiki.ros.org/ROS/Introduction>>.

\_\_\_\_\_. **Gazebo Simulator**. 2022. Accessed: 2025-05-24. Disponível em: <<https://gazebosim.org/home>>.

\_\_\_\_\_. **TurtleBot**. 2025. Accessed: 2025-05-24. Disponível em: <<https://www.turtlebot.com/>>.

GAO, X. et al. Design and ground performance evaluation of a multi-joint wheel-track composite mobile robot for enhanced terrain adaptability. **Applied Sciences**, v. 13, n. 12, 2023. ISSN 2076-3417. Disponível em: <<https://www.mdpi.com/2076-3417/13/12/7270>>.

INC., D. **Docker**. 2025. Accessed: 2025-05-25. Disponível em: <<https://docs.docker.com/get-started/docker-overview/>>.

ROBOTIS. **TurtleBot3 e-Manual: Overview**. 2025. Accessed: 2025-05-24. Disponível em: <<https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/>>.

ROBOTIS-GIT. **TurtleBot3 GitHub Repository**. 2025. Accessed: 2025-05-24. Disponível em: <<https://github.com/ROBOTIS-GIT/turtlebot3>>.

SOBIERAJ, M.; KOTYŃSKI, D. Docker performance evaluation across operating systems. **Applied Sciences**, v. 14, n. 15, 2024. ISSN 2076-3417. Disponível em: <<https://www.mdpi.com/2076-3417/14/15/6672>>.

WEN, L. et al. Bare-metal vs. hypervisors and containers: Performance evaluation of virtualization technologies for software-defined vehicles. In: . [s.n.], 2023. Disponível em: <[https://www.researchgate.net/publication/372496789\\_Bare-Metal\\_vs\\_Hypervisors\\_and\\_Containers\\_Performance\\_Evaluation\\_of\\_Virtualization\\_Technologies\\_for\\_Software-Defined\\_Vehicles](https://www.researchgate.net/publication/372496789_Bare-Metal_vs_Hypervisors_and_Containers_Performance_Evaluation_of_Virtualization_Technologies_for_Software-Defined_Vehicles)>.

