[Introduction 2](#_Toc1)

[Background and motivation 2](#_Toc2)

[Research questions and objectives 2](#_Toc3)

[Scope and limitations 2](#_Toc4)

[Literature Review 3](#_Toc5)

[Overview of ROS1 and ROS2 3](#_Toc6)

[Descrizione delle parti di ROS2 3](#_Toc7)

[La strumentazione offerta in ROS 4](#_Toc8)

[Da ROS 1 a ROS 2 4](#_Toc9)

[Introduction to Navigation Stack and Google Cartographer 6](#_Toc10)

[Review of related work on porting navigation stack from ROS1 to ROS2 6](#_Toc11)

[Overview of open source fleet management systems, particularly Open-RMF 6](#_Toc12)

[Methodology 6](#_Toc13)

[Description of research platform and hardware used 6](#_Toc14)

[Steps taken to port the navigation stack from ROS1 to ROS2 6](#_Toc15)

[Hardware adaptation process from Rover Zero to Rover Mini 6](#_Toc16)

[Integration of the navigation stack with Open-RMF 6](#_Toc17)

[Testing and evaluation of the system 6](#_Toc18)

[Results and Analysis 6](#_Toc19)

[Presentation of results from testing and evaluation 6](#_Toc20)

[Comparison of system performance before and after the porting process 6](#_Toc21)

[Discussion on the challenges encountered and how they were addressed 6](#_Toc22)

[Conclusion and Future Work 6](#_Toc23)

[Summary of the research findings 6](#_Toc24)

[Contributions to the field of electrical engineering and robotics 6](#_Toc25)

[Recommendations for future work and improvements 6](#_Toc26)

[References [[references]] 7](#_Toc27)

[List of sources cited in the thesis 7](#_Toc28)

[Appendices 7](#_Toc29)

[Additional technical details and documentation 7](#_Toc30)

# Introduction

## Background and motivation

Elettra è una ente di ricerca internazionale che fornisce due forme di sorgenti di radiazioni (light source): il sincrotrone Elettra ed il laser a elettroni liberi fermi.

https://www.elettra.eu/

Essendo un grande ente di ricerca, si sviluppano applicazioni che posso rendere migliore lo svolgimento delle attività all’interno dello stesso. Uno di questi miglioramenti è l’utilizzo della robotica nelle attività ordinarie, ad esempio trasporto di oggetti, interazioni con gli utenti che utilizzano le macchine, ... Tale moltitudine di applicazioni ha reso l’ambiente ricco di una varietà di robot da diversi produttori ed un problema che si presenta con i diversi produttori è l’interazione in maniera intelligente tra di essi. In questi ultimi anni, dalla open robotics hanno sviluppato Open-rmf[[1]](#footnote-2) un gestore di robot multipiattaforma “multi fleet”.

Inoltre è stato sviluppato, attraverso il Robotics and Remotization initiative, un planner di navigazione D-Star-Based Optimized Trajectory Planner[[2]](#footnote-3) per ROS1 ottimizzato per la navigazione in ambienti industriali grandi e complessi come quello di elettra che si vuole appilcare ad un robot il mini della roverrobotics[[3]](#footnote-4)

## Research questions and objectives

## Scope and limitations

Lo scopo di questa tesi è aggiornare il sistema di planning sviluppato portandolo da ROS1 a ROS2[[4]](#footnote-5) utilizzando la struttura rinnovata di NAV2 rispetto a NAV1, applicarlo al nostro robot MINI della rover robotics ed inserirlo dentro l’ambiente OPEN-RMF che accoglierà in futuro tutte le altre famiglie di robot presenti in azienda.

Le uniche limitazioni che al momento non abbiamo un controllo diretto agli ascensori e non permettono di testare tutte le funzionalità offerte da OPEN\_RMF.

# Literature Review

Adesso descriveremo l’insieme delle tecnologie usate

## Overview of ROS1 and ROS2

Il Robot Operating System[[5]](#footnote-6) (ROS) è un insieme di librerie e strumenti open-source che aiutano a sviluppare applicazione in ambito robotico. Diversamente dal nome non è un sistema operativo ma un software development kit (SDK) che ti fornisce tutti gli strumenti per creare il tuo robot fornendo alcune funzionalità tipiche di un OS: hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management.

I principali vantaggi che offre ROS sono:

* un sistema di comunicazione message-passing system chiamato “middleware” or “plumbing”. Che permette la comunicazione tra i “nodi” del sistema con un anonymous publish/subscribe pattern permettendo fault isolation, separation of concerns and clear interfaces e un sistema synchronous RPC-style communication tra i services
* Sistemi di controllo del software come launch, introspection, debugging, visualization, plotting, logging, and playback. Che migliorano gli strumenti per lo sviluppo.
* provides tools and libraries for obtaining, building, writing, and running code across multiple computers
* Una communità attiva e diffusa con al centro la Open Robotics[[6]](#footnote-7) che mantiene il codice e hosta le release

### Descrizione delle parti di ROS2

La documentazione[[7]](#footnote-8) fornisce una descrizione esaustiva di tutti gli strumenti base offerti da ROS2, alcune funzionalità qui descritte sono solo disponibili in ROS2 e non in ROS1:

* The **ROS graph** è una peer-to-peer network of processes di ROS2 che elaborano dati simultaneamente. Comprende tutti gli eseguibili e le connessioni tra di essi.
* **ROS node** è il processo che dovrebbe realizzare un singolo obbiettivo e dovrebbe essere realizzato in maniera modulare in modo da essere facilmente interfacciabile da alti nodi che descrivono altri specifici funzionalità che può essere concatenato all’altro nodo. Ogni nodo comunica con gli altri nodi con “topics, services, actions, or parameters”.
* **Topics** sono un elemento fondamentale del grafo ROS che funge da bus per consentire ai nodi di scambiarsi messaggi.
* **Services** si basano su un modello di richiesta e risposta, a differenza del modello publisher-subscriber dei topic. Mentre i topic consentono ai nodi di iscriversi a flussi di dati e ricevere aggiornamenti continui, i Services forniscono dati solo quando vengono specificamente richiesti da un client.
* A **parameter** sono i parametri di configurazione dei nodi che permettono di modificarli senza cambiare il codice sorgente
* **Actions** sono stati sviluppati per task lunghi e consistono di 3 parti: a goal,a feedback, and a result. Sono costruiti al di sopra dei topic e dei servizi, la funzionalità è simile hai services ma può esser fermato e fornisce un update continuo sullo stato dell’azione.

### La strumentazione offerta in ROS

Qui di seguito faccio un elenco degli strumenti offerti da ROS per lo sviluppo di ogni applicazione e con una breve descrizione a riguardo:

* **Launch** files struttura che ti permettono di configurare ed eseguire multipli nodi ROS simultaneamente.
* **rqt** is a graphical user interface (GUI) tool for ROS 2, fornisce dei plugin che possono essere aggiunti all’interfaccia per facilitare lo sviluppo, il monitoraggio e la diagnostica del sistema ROS
* **rqt\_console** (log visualizer) è un visualizzatore di log di ros2
* Interazione con diversi **simulatori** come Gazebo(anch’esso un progetto della open robotics) , Webots, etc.
* **RViz** is a 3D visualizer for the Robot Operating System (ROS) framework.
* **rosdep** is a dependency management utility that can work with packages and external libraries.
* The **sros2** package provides the tools and instructions to use ROS 2 on top of DDS-Security.

### Da ROS 1 a ROS 2

ROS è nato per uso accademico per il controllo di un robot ma si è diffuso anche in ambito commerciale nel quale ha riscontrato nuovi ambienti non pensati per l’uso accademico. I nuovi casi sono i seguenti:

* La gestione di gruppi di robot
* Sviluppo su piccole schede embedded
* Sistemi Real Time
* Network non ideali
* Orientarsi anche in ambito industriale
* Sviluppo di patterns e strumenti per la costruzione di sistemi

Tutto questo sviluppo ha portato allo sviluppo di ROS 2, maggiori informazioni sono disponibili nella documentazione di sviluppo[[8]](#footnote-9).

Questo ha portato all’abbandono di ROS1 che conferma l’EOL per maggio 2025.

## Introduction to NAV2 and Google Cartographer

Non solo abbiamo il passaggio da ROS 1 a ROS 2 ma anche abbiamo che il vecchio ROS Navigation Stack è stato sostituito da Nav2[[9]](#footnote-10) in sviluppo sotto la Open Navigation[[10]](#footnote-11).

allows for mobile robots to navigate through complex environments to complete user-defined application tasks with nearly any class of robot kinematics. Not only can it move from Point A to Point B, but it can have intermediary poses, and represent other types of tasks like object following, complete coverage navigation, and more

It provides perception, planning, control, localization, visualization, and much more to build highly reliable autonomous systems. This will compute an environmental model from sensor and semantic data, dynamically path plan, compute velocities for motors, avoid obstacles, and structure higher-level robot behaviors

Nav2 uses behavior trees to create customized and intelligent navigation behavior via orchestrating many independent modular servers. A task server can be used to compute a path, control effort, behavior, or any other navigation related task. These separate servers communicate with the behavior tree (BT) over a ROS interface such as an action server or service. A robot may utilize potentially many different behavior trees to allow a robot to perform many types of unique and complex tasks.

The expected inputs to Nav2 are TF transformations conforming to REP-105, a map source if utilizing the Static Costmap Layer, a BT XML file, and any relevant sensor data sources. It will then provide valid velocity commands for the motors of a holonomic or non-holonomic robot to follow when properly configured.

## Review of related work on porting navigation stack from ROS1 to ROS2

## Overview of open source fleet management systems, particularly Open-RMF

# Methodology

## Description of research platform and hardware used

## Steps taken to port the navigation stack from ROS1 to ROS2

## Hardware adaptation process from Rover Zero to Rover Mini

## Integration of the navigation stack with Open-RMF

## Testing and evaluation of the system

# Results and Analysis

## Presentation of results from testing and evaluation

## Comparison of system performance before and after the porting process

## Discussion on the challenges encountered and how they were addressed

# Conclusion and Future Work

## Summary of the research findings

## Contributions to the field of electrical engineering and robotics

## Recommendations for future work and improvements

# References [[references]]

## List of sources cited in the thesis

# Appendices

## Additional technical details and documentation

1. <https://www.open-rmf.org/> [↑](#footnote-ref-2)
2. <https://doi.org/10.1007/978-3-031-45770-8_29> [↑](#footnote-ref-3)
3. https://roverrobotics.com/ [↑](#footnote-ref-4)
4. S. Macenski, T. Foote, B. Gerkey, C. Lalancette, W. Woodall, “Robot Operating System 2: Design, architecture, and uses in the wild,” Science Robotics vol. 7, May 2022.

   @article{

   doi:10.1126/scirobotics.abm6074,

   author = {Steven Macenski and Tully Foote and Brian Gerkey and Chris Lalancette and William Woodall },

   title = {Robot Operating System 2: Design, architecture, and uses in the wild},

   journal = {Science Robotics},

   volume = {7},

   number = {66},

   pages = {eabm6074},

   year = {2022},

   doi = {10.1126/scirobotics.abm6074},

   URL = {https://www.science.org/doi/abs/10.1126/scirobotics.abm6074}

   } [↑](#footnote-ref-5)
5. <https://www.ros.org/> [↑](#footnote-ref-6)
6. <https://www.openrobotics.org/> [↑](#footnote-ref-7)
7. <https://docs.ros.org/en/rolling/index.html> [↑](#footnote-ref-8)
8. <https://design.ros2.org/> [↑](#footnote-ref-9)
9. <https://docs.nav2.org/> [↑](#footnote-ref-10)
10. <https://www.opennav.org/> [↑](#footnote-ref-11)