Metra SCITOS A5

Autonomous Navigation Manual

Project head: Prof. Dr.-Ing. Pascal Meißner

Authors & Project members:

- Kubegenov Yernar
- Varma Surajkumar Dineshbhai
- Mutallimov Fikrat

Date: 25th December 2023

Contents

1 Robot Hardware Details

1.1 Control System

1.1.1 Description

The robot's control system comprises two main parts: the base logic and a standard computer.

- Base logic manages motors, bumpers, sensors (magnetic, laser, Kinect), and power supply for all components.
- Safety features are integrated, automatically shutting down the system during collisions to prevent damage.

1.1.2 Computer Specifications

Utilizes an Intel Core i7-6700T processor, facilitating communication and control through a CAN-Bus connection.

1.2 Sensors

1.2.1 Sensor Details

- Microsoft Kinect: Scans the environment using RGB and depth perception.
- Asus Xtion Pro Live: Equipped with RGB, depth perception, and a microphone.
- Front and rear lasers: Measure distance within a specific range.
- RFID reader: Detects cards/tokens, usage depends on the application.
- Bumpers and magnet sensor: Identifies collisions and tracks magnetic tape on the floor.

1.3 Kinect

1.3.1 Mounting and Control

- Positioned on the robot's head, the Microsoft Kinect V2 lacks internal motor control.
- Controlled by Flir motion control motors, managed within the MIRA-Commercial package.

1.3.2 Compatibility and Data Retrieval:

- Microsoft doesn't offer Linux support, but the open-source OpenKinect project provides libFreenect2 for data retrieval.
- Data processing capabilities using OpenCV, a library for image processing and machine vision.

1.4 Interfaces

1.4.1 Charging Options

- Two charging methods available:
- Rear-side plug for direct charging.
- Contacts on the robot's underside for automated charging via a separate station.

1.4.2 Connectivity

- External USB port allows connections with additional devices.

1.4.3 User Interaction

- Interaction facilitated through a 14" touch display featuring integrated speakers and a resolution of 1024x768 pixels.

2 Robot Software Details

2.1 Operating System (OS)

- The robot runs on Linux Ubuntu 16.04 Long Term Support (LTS) with a Unity Desktop.
- This LTS version gets updates until April 2021.
- The manufacturer installed the OS on the robot.

2.2 ROS (Robot Operating System)

- ROS is a framework for controlling personal robots under the Berkeley Software Distribution (BSD) license.
- It consists of different parts:
  - Hardware abstraction
  - Device drivers
  - Communication between modules
  - Package management
  - Program libraries for managing software on different systems.
- ROS offers space for personal modules and has a supportive community.
- Projects using the Scitos G5 robot are supported, making basic functionalities relatively easy to use.

2.3 MIRA Software Framework

- MIRA is a modular software framework by MetraLabs and Ilmenau University of Technology.
- It includes a graphical user interface and modules to control sensors and the robot.
- Licensed under GNU General Public License (GPL) v3.0 and other licenses for specific algorithms.
- It operates based on "units" and "channels" where units provide functionalities and are connected through a global bus system.
- MIRAcenter is a user-friendly graphical interface showing channels, data, and unit statuses.
- MIRAtape records and plays back chosen channels in real-time, enabling simulation without direct hardware access.

2.3.1 Features of MIRA's User Interface (MIRAcenter):

- Visualizes channel data and unit statuses.
- Displays units' statuses (green for normal, yellow for lagging, red for errors).
- Offers data visualization and playback controls for analysis.

2.3.2 MIRAtape Tool Features:

- Records chosen channels in real-time for playback and simulation purposes.
- Enables editing and management of recorded data channels.
- Allows playback of recorded data for testing and benchmarking algorithms.

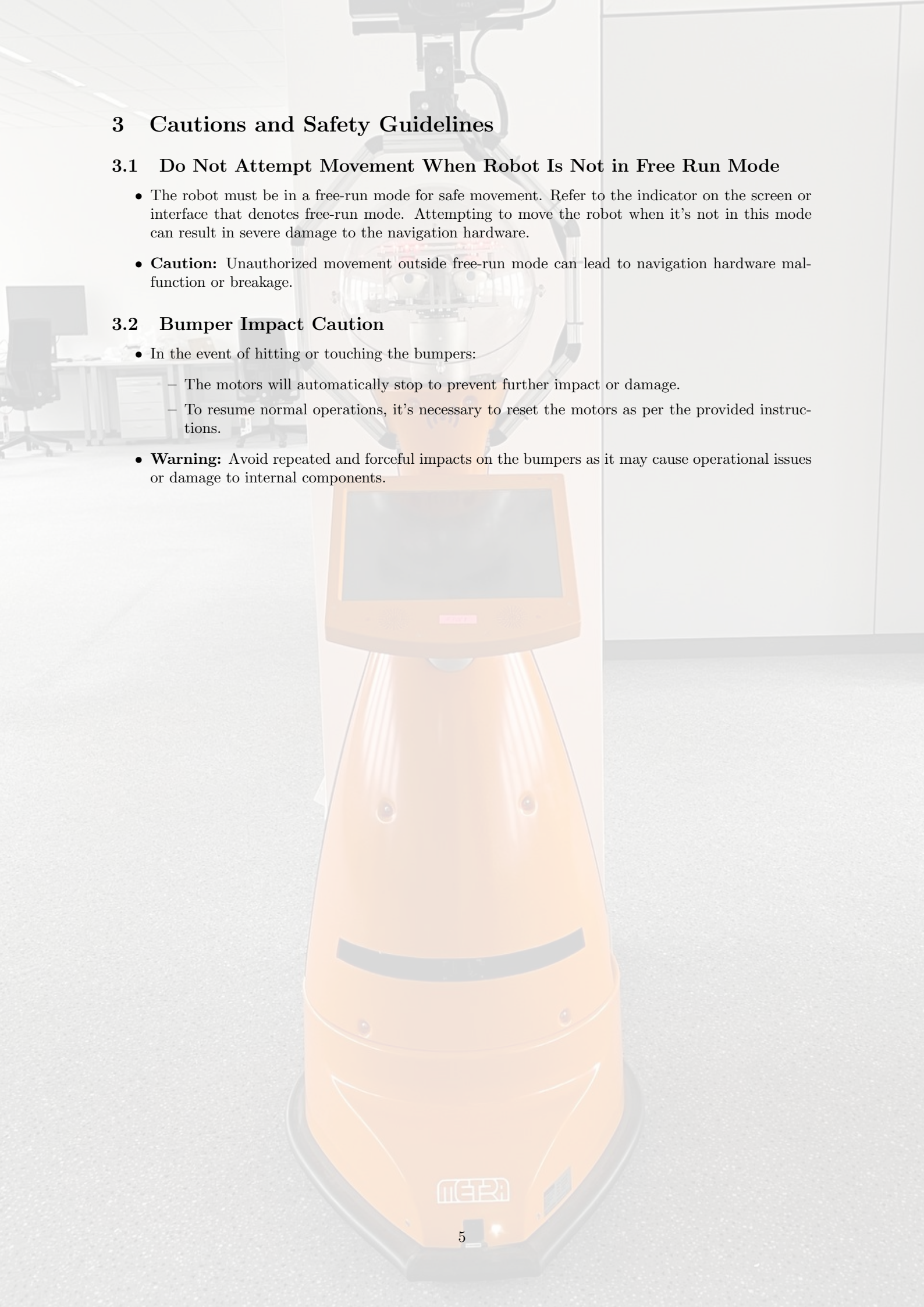
3 Cautions and Safety Guidelines

3.1 Do Not Attempt Movement When Robot Is Not in Free Run Mode

- The robot must be in a free-run mode for safe movement. Refer to the indicator on the screen or interface that denotes free-run mode. Attempting to move the robot when it's not in this mode can result in severe damage to the navigation hardware.
- **Caution:** Unauthorized movement outside free-run mode can lead to navigation hardware malfunction or breakage.

3.2 Bumper Impact Caution

- In the event of hitting or touching the bumpers:
  - The motors will automatically stop to prevent further impact or damage.
  - To resume normal operations, it's necessary to reset the motors as per the provided instructions.
- **Warning:** Avoid repeated and forceful impacts on the bumpers as it may cause operational issues or damage to internal components.



4 Steps to Turn Power ON the Robot

1. **Locate the Special Key:**
   - The key is located at the bottom near a small screen. Refer to the image given below for visual guidance.

Figure 1: Turn on the key

   - Turn the key to the right to power on the robot's hardware and EC components.
2. **Navigation using the Small White Coloured Knob:**
   - Use the small white coloured knob located next to the small screen for navigation through menus.

Figure 2: Use this white knobe

   - Rotate the knob to navigate through the menus displayed on the screen.
   - Once the desired menu is highlighted, press the knob to execute the selected option.

3. **Red Coloured Emergency Button:**
    - Locate the red-coloured emergency button designed to halt the robot's motion.
    - In case of an emergency or to stop the robot's movement, press the red button.

Figure 3: Use this Red button

    - To release the motor brakes after stopping, pull back the emergency button.
4. **System Movements during Booting:**
    - During the boot-up process, you may notice movements in components like the head and cameras.
    - These movements signify system initialization and are entirely normal. They indicate that the system is undergoing its startup routine.
5. **User Account Information:**
    - You may encounter two user accounts during the startup. The main user is "scitosrobo," and its password is "SCITOS" (in lowercase).

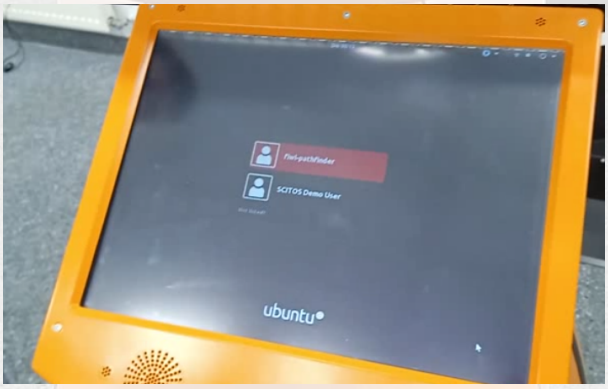


Figure 4: Select appropriate.

6. **Wait for Initialization:**
    - Wait for the welcome display to appear and select "Start robot and PC" during initialization.
    - Press "Enter" to continue during the loading phase.
7. **Wi-Fi Connectivity:**
    - Ensure Wi-Fi connectivity; connect to available public networks or hidden networks with specific network names and security details.
8. **Initiate Robot Movements:**
    - Use the Mira framework or ROS 2 framework to initiate and control the robot's movements.

5 Operate the Robot using the “Mira Framework”

1. **Booting Up:**

   - Wait for the system to start completely. You’ll see a graphical interface (GUI) of Linux version 22.04 Ubuntu.

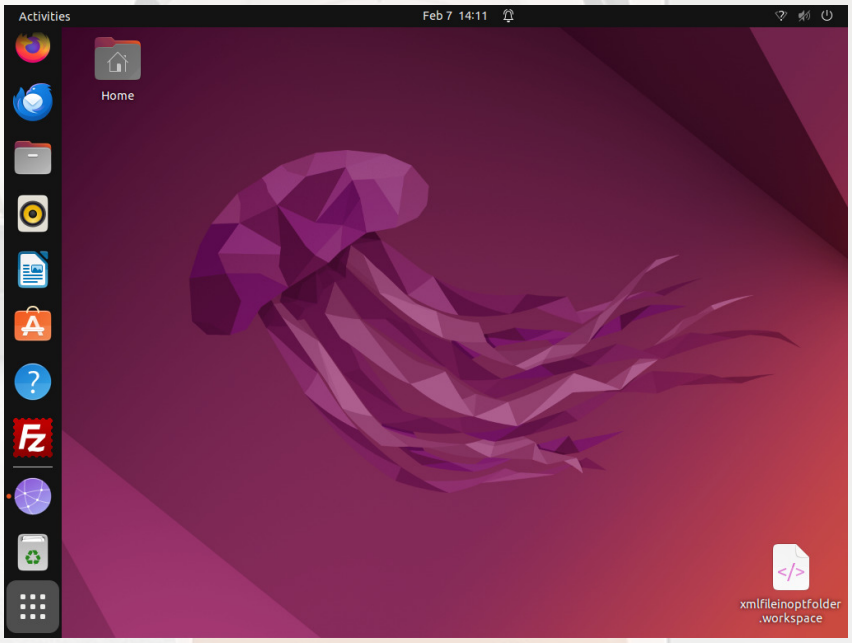


Figure 5: Home Screen

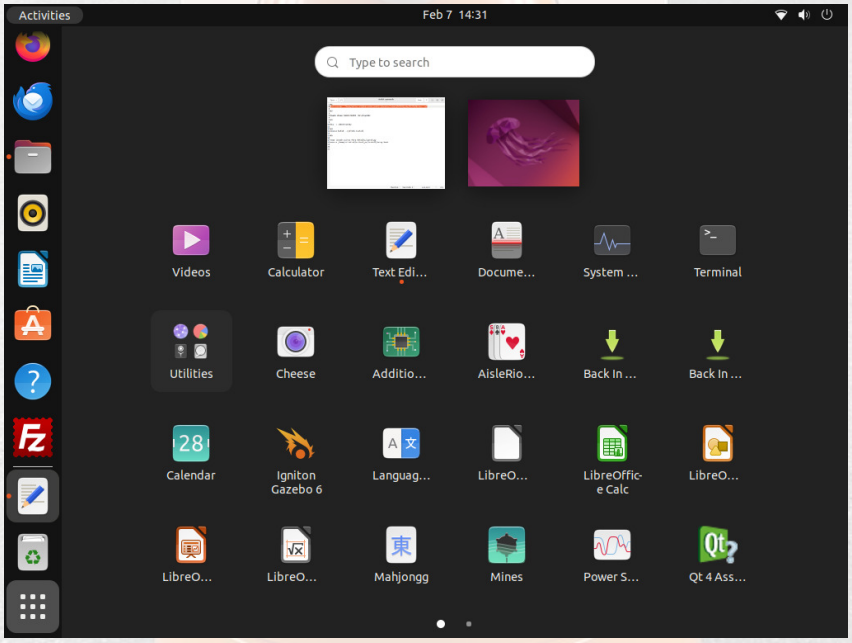


Figure 6: Available apps & Programs

2. **Accessing Necessary Commands:**

   - Go to the ’Home’ screen. Look for the file named ”useful commands.” Inside this file, find the first command labeled as ”$miracenter /home/scitos-a5/MetraLAbs-public/domains/robot/SCITOS/etc/SCIT Copy this command.

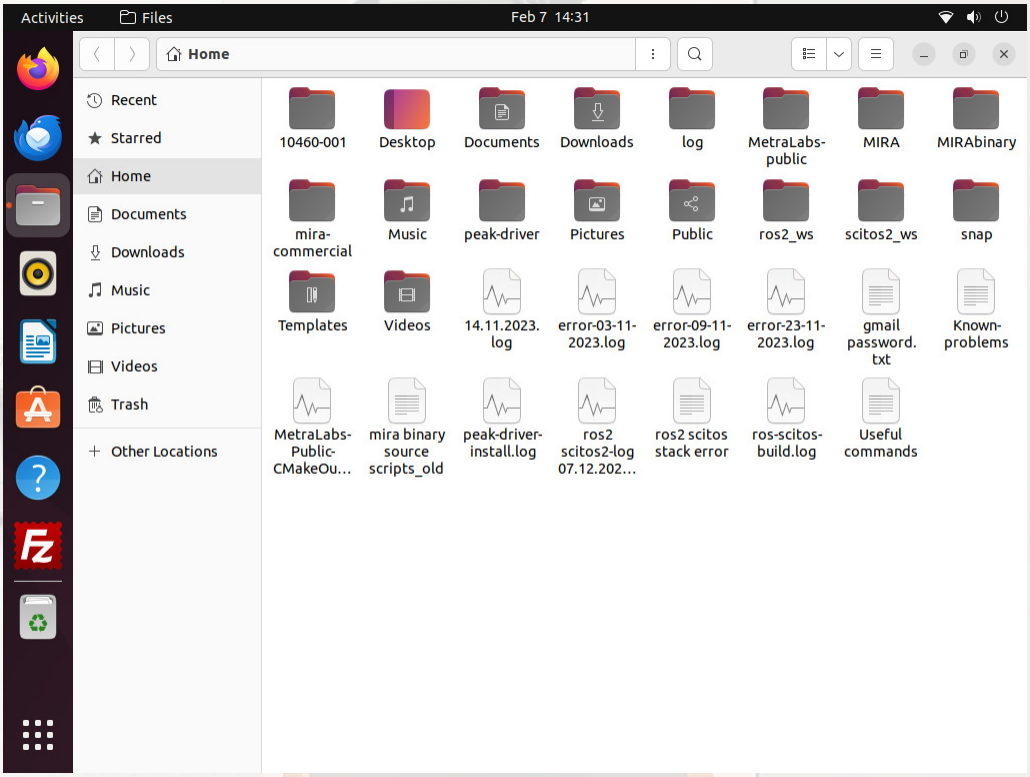


Figure 7: Select "Useful commands"

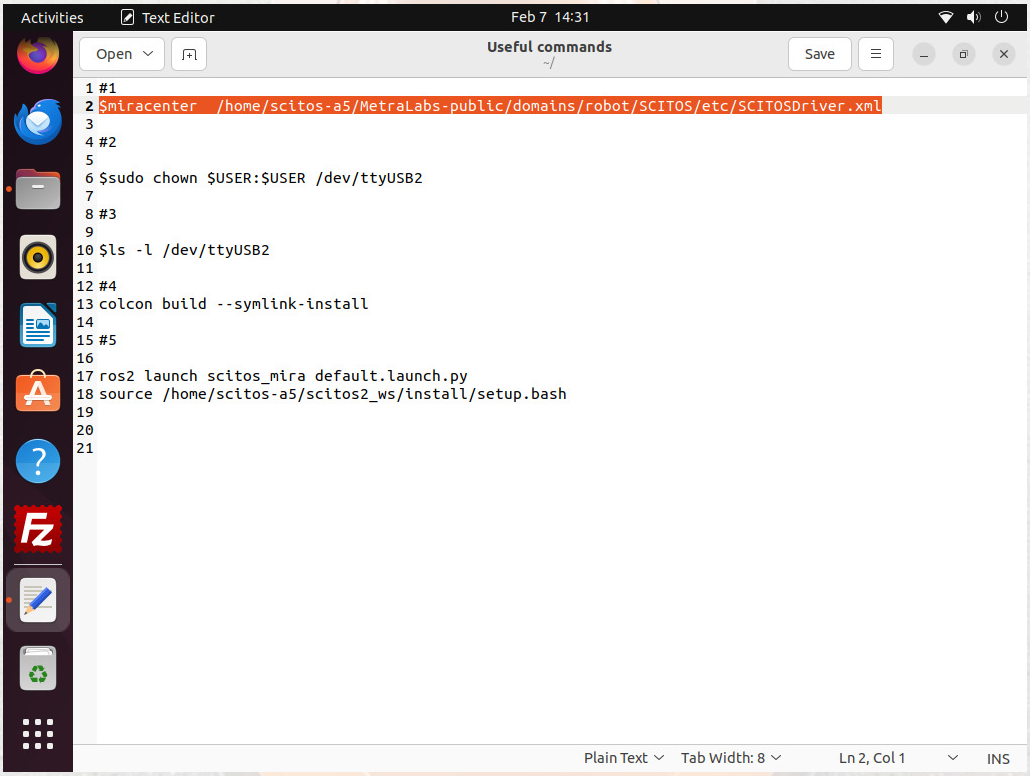


Figure 8: copy the selected.

3. **Opening Terminal and Executing Command:**

   - Press "Ctrl+Alt+T" to open the terminal. Paste the copied command and press "Enter." Warnings may appear about different hardware components using default settings. Look for the "Booting up" message; this indicates that the Mira Center GUI for robot controls has launched.

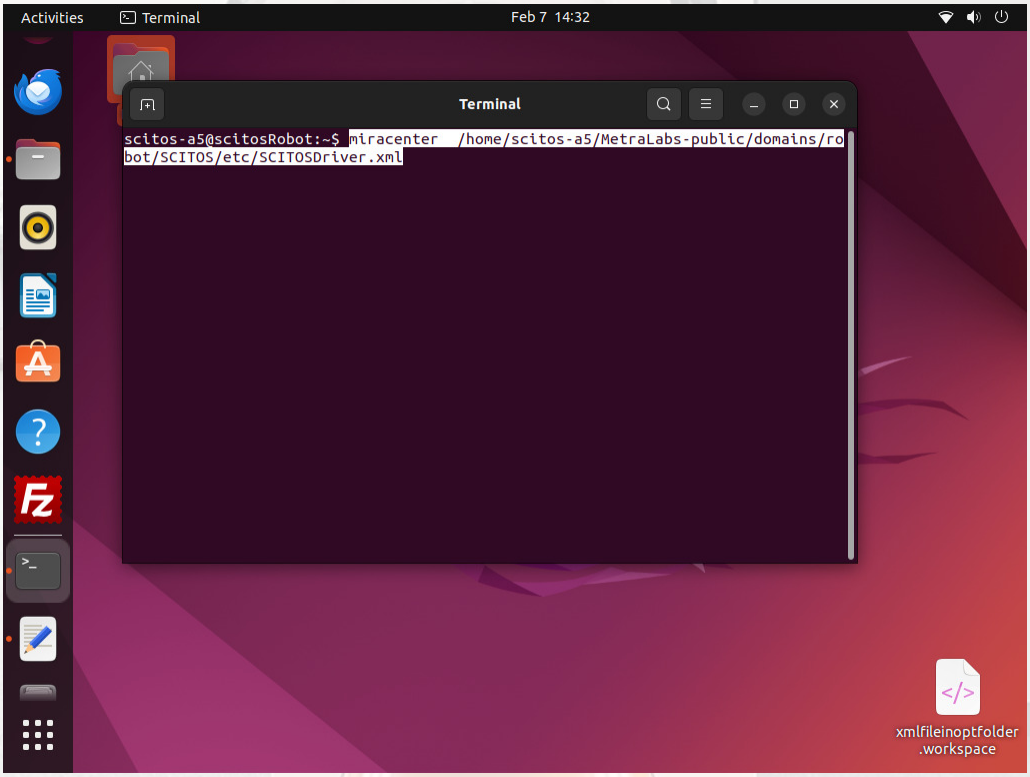


Figure 9: Paste in Terminal

4. **Navigating the Mira Center GUI:**

   - Close the terminal once the Mira Center is running. Use the small GUI displayed on the bottom left of the screen for robot control. This GUI has four arrow buttons for movement and a central red "stop" button.

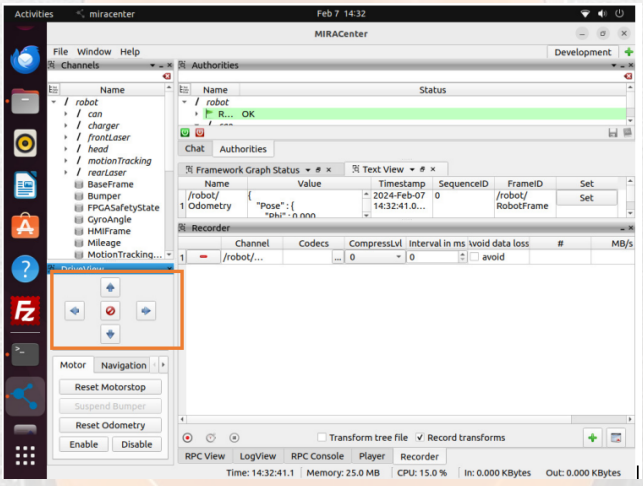


Figure 10: Mira Center GUI

5. **Resetting Motors for Motion:**

- Reset the motors if they are locked due to security bumper sensors. Click on the "Motor" menu and select the "Reset Motor" option. Imagine the main screen as the robot's face.

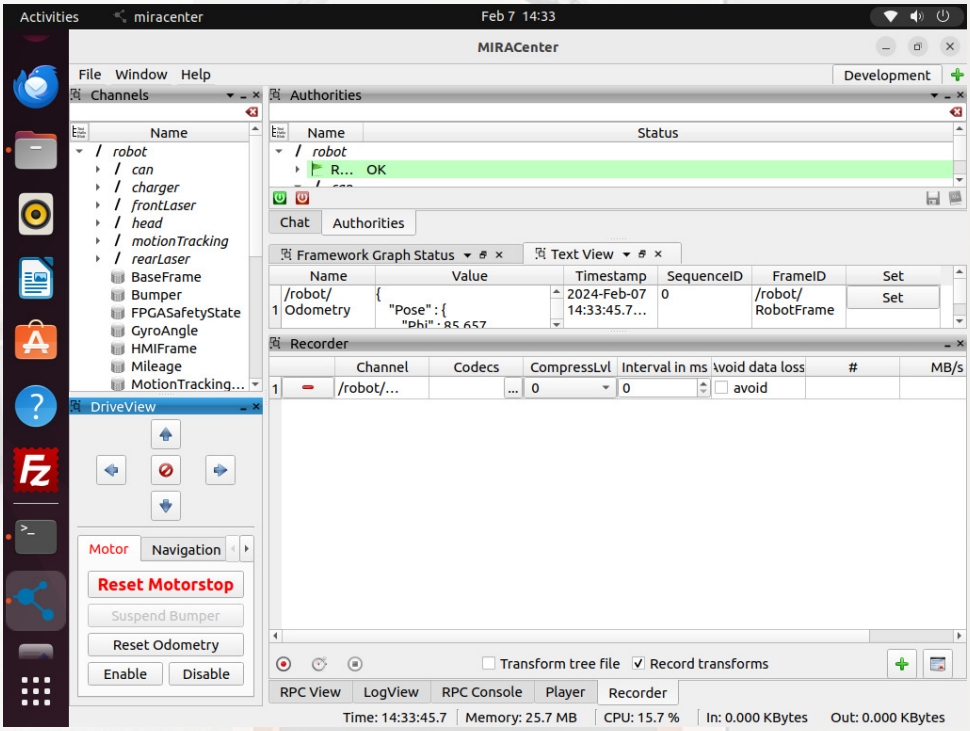


Figure 11: 'Red sign' when motors are locked

6. **Controlling Robot Movement:**

- To move the robot, place the mouse pointer on the desired arrow button. Press the upward arrow for backward movement and the downward arrow for forward motion. You don't need to continuously hold the arrow; once pressed, it will keep moving until you hit the stop button, or an external factor stops it.

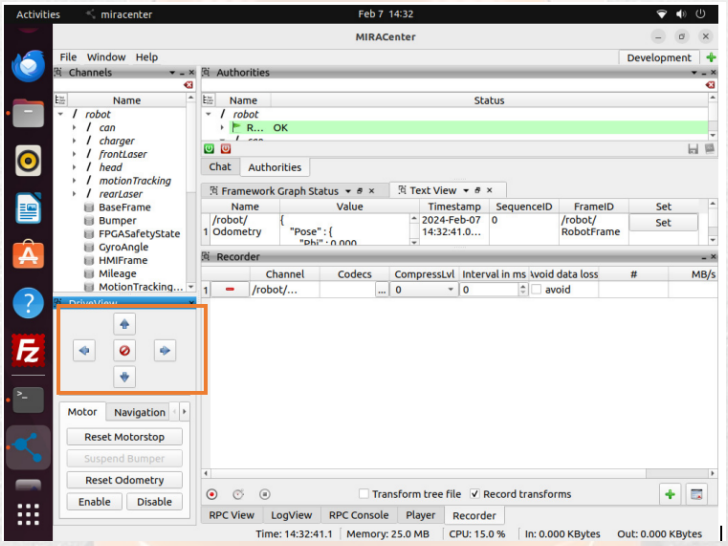


Figure 12: Use blue arrows

7. **Emergency Stop and Directional Movements:**

- If needed, use the red emergency button to halt the robot's motion, as explained in the caution section. You can also guide the robot's direction while it's in motion by clicking the right or

left arrow buttons. Holding down any arrow button will make the robot move in a circular motion in that direction.

8. **Adjusting Robot's Velocity:**

   - Pressing any(forward/backward) arrow key three times will increase the robot's speed. You'll notice changes in the "phi" pose value during motion.

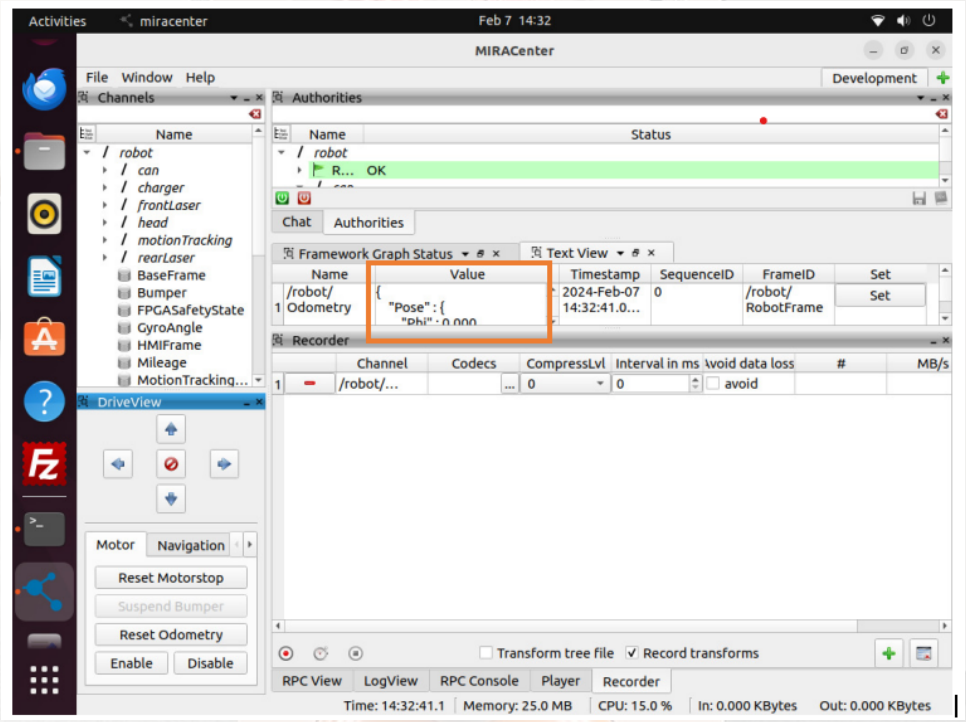


Figure 13: Changing scale during motion

9. **Closing the Motion Program:**

   - After interacting with the robot's motion, simply click on the close button(top right corner) of the motion program to conclude.

6 Operating the Robot with ROS2 Framework

1. **Booting Up the Operating System**

   - Ensure successful boot-up of the OS before proceeding with robot navigation.

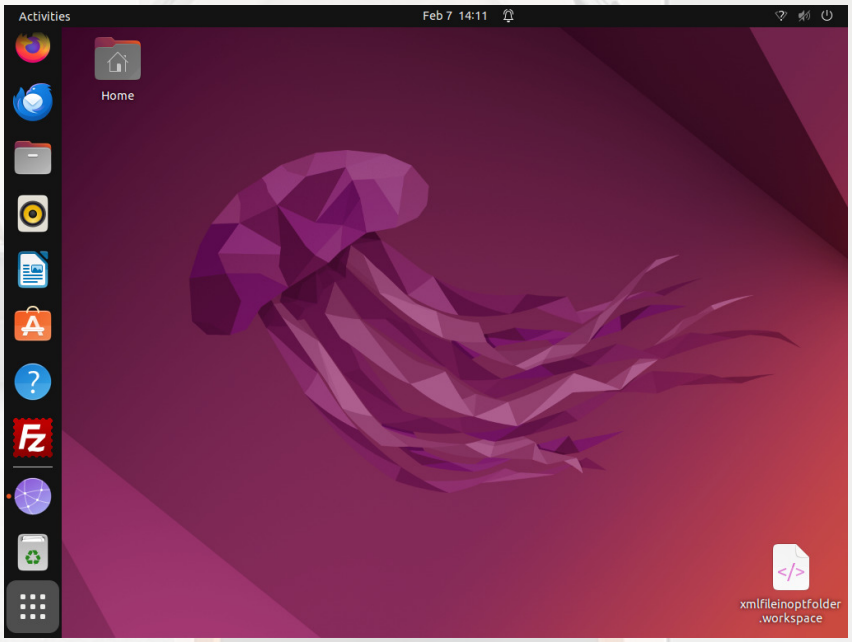


Figure 14: Home Screen

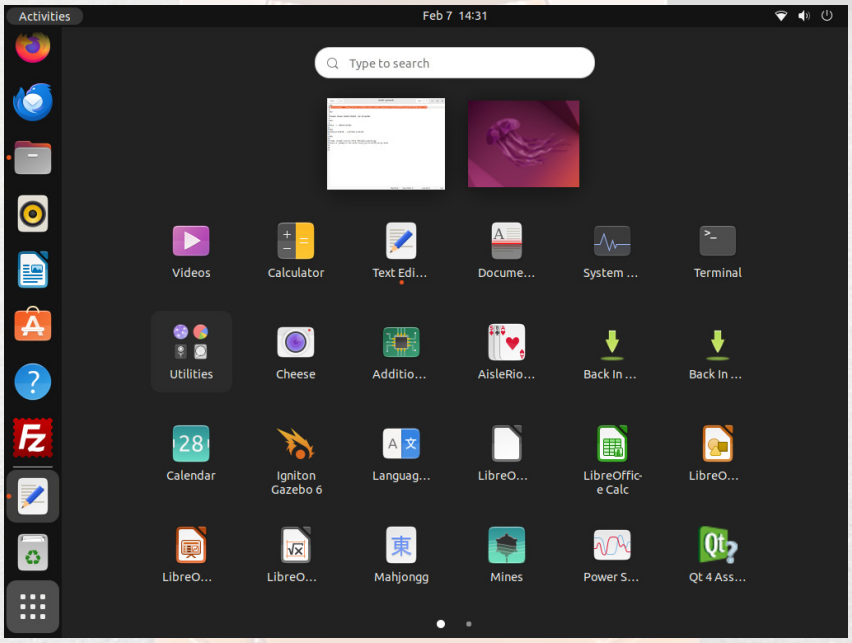


Figure 15: Available apps & Programs

2. **Accessing Necessary Commands**

   - Navigate to the 'Home' screen and locate the file named "useful commands." Within this file, various commands are available. Look for the specific command: "ros2 launch scitos_mira default.launch.py."

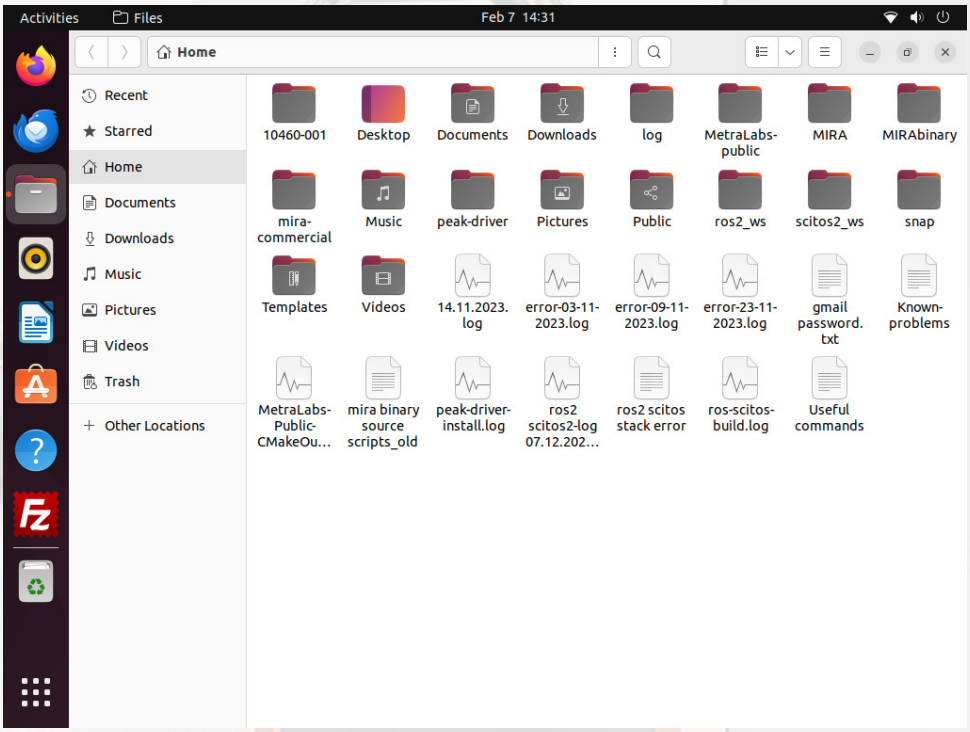


Figure 16: Select "Useful commands"

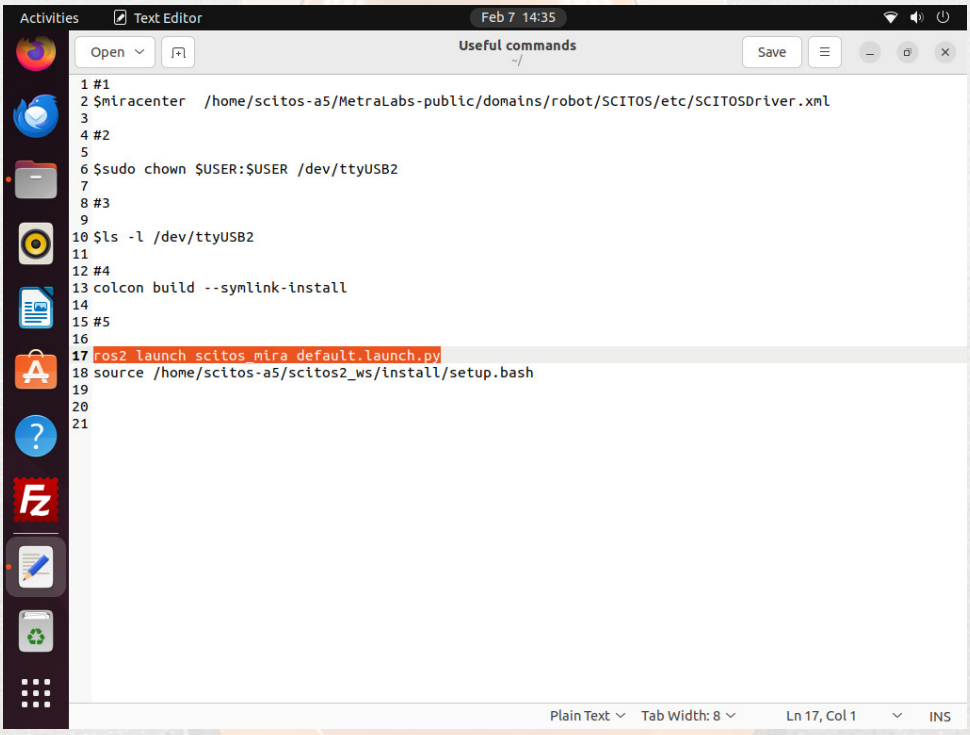


Figure 17: copy the selected.

3. **Launching the ROS2 Tool**

- Copy the identified command and open the terminal by pressing "Ctrl + Alt + T." Paste the command into the terminal and press "Enter." This command initializes the tool required for robot navigation using the ROS2 framework.

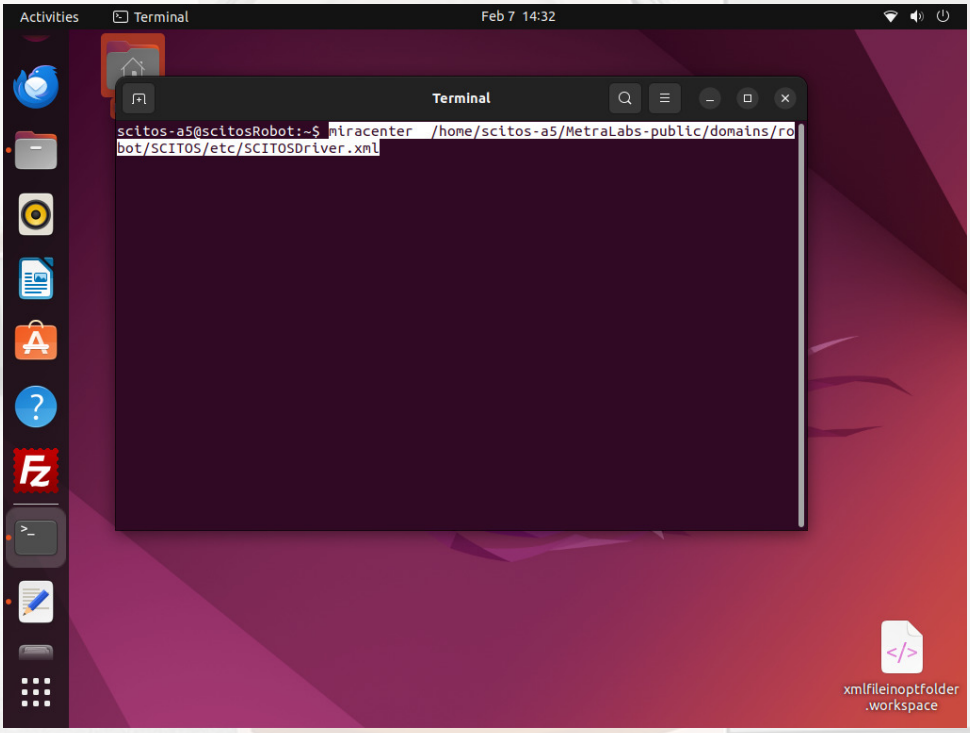


Figure 18: Paste in Terminal

4. **Understanding the Graphical User Interface (GUI)**

- Upon executing the command, a GUI interface will appear.If not , another way to access this interface is by using the command "rqt" in the terminal.

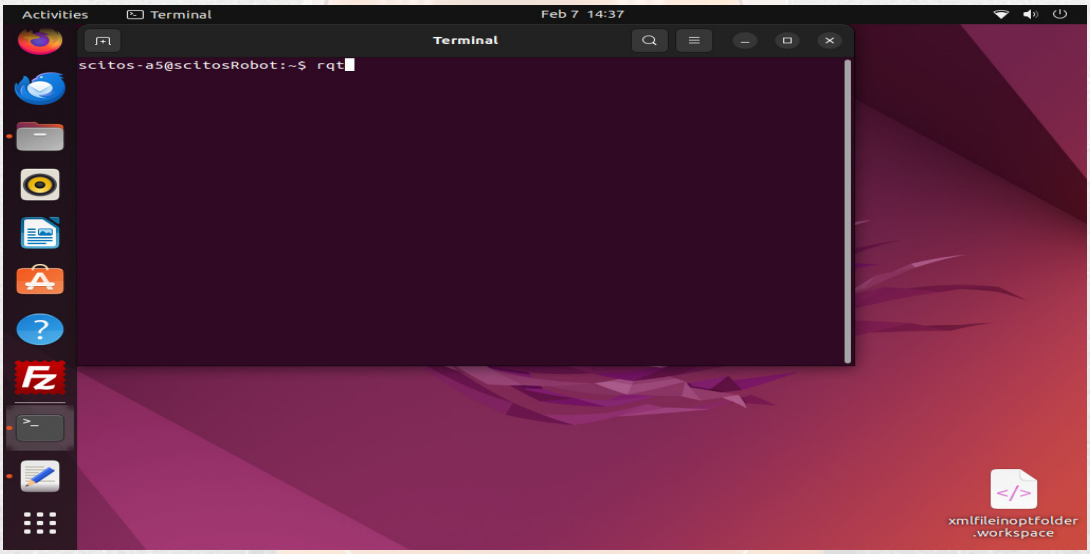


Figure 19: Use 'rqt'

5. **GUI Navigation Controls and Information**

- The GUI features sliders for navigation control. The horizontal slider facilitates left and right movements, while the vertical slider manages forward and backward directions. Each axis slider at zero signifies the robot's stopped motion in that direction.

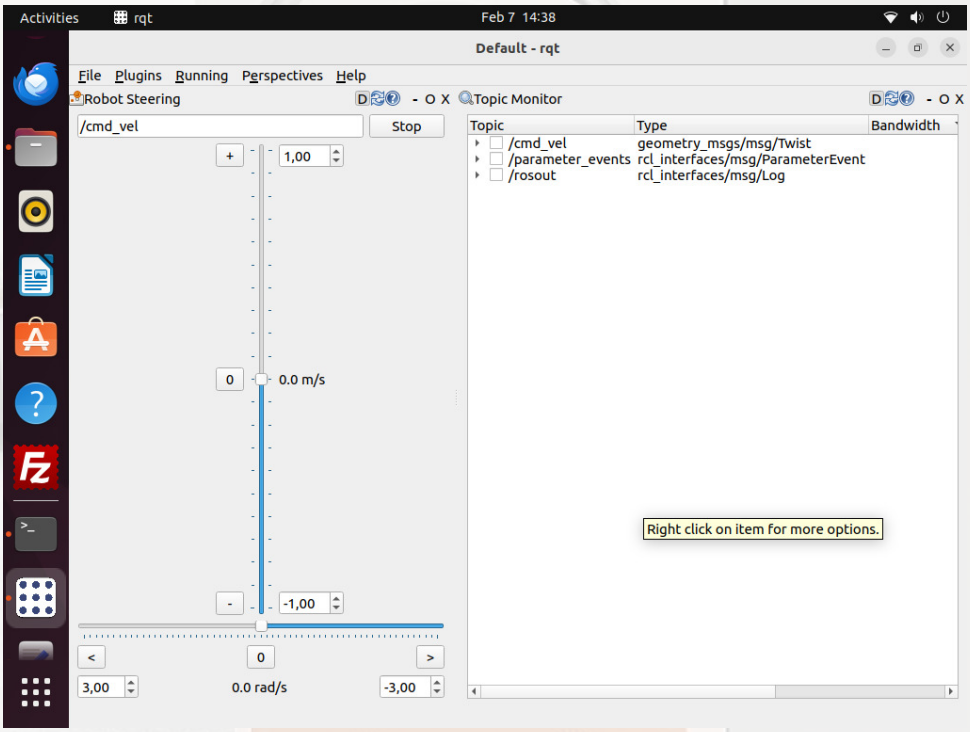


Figure 20: GUI to navigate via ROS

- On the right side of the interface, detailed information about the robot's status and functionalities is displayed, providing insights that users can explore further.

6. **Exploring Information and Returning to Drive Menu**

- To delve into information and return to the drive menu, navigate to "Plugins" in the menu bar. Select "Robot Tools" and click on "Robot Steering." This action will take you back to the GUI, providing controls to drive the robot using sliders.

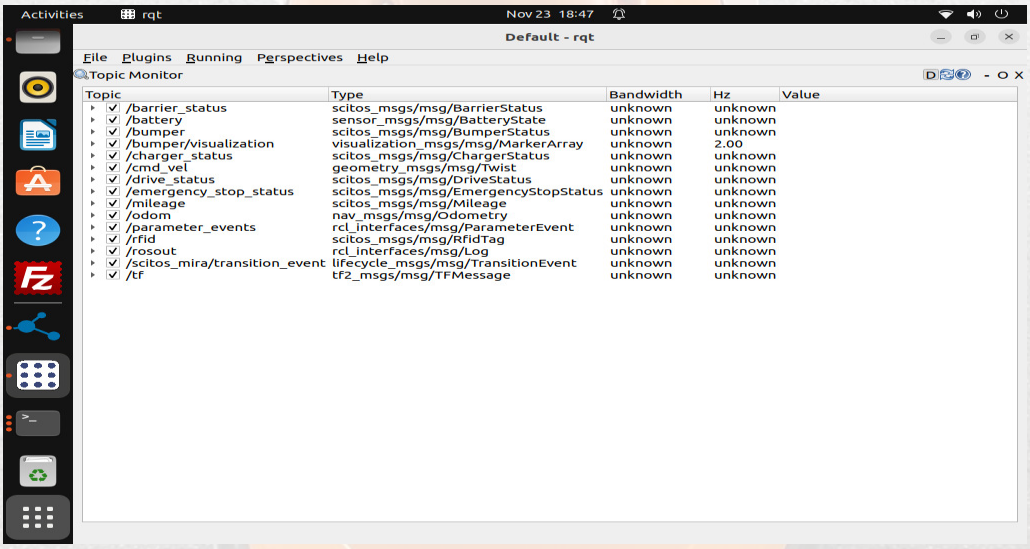


Figure 21: Plugin menus.

7. **Utilizing Sliders for Robot Motion**

- Consider the screen as the robot's "face." To command backward movement, slide the vertical slider upwards. For forward motion, slide it downwards. Setting the sliders to zero halts the robot's motion in the respective axis.

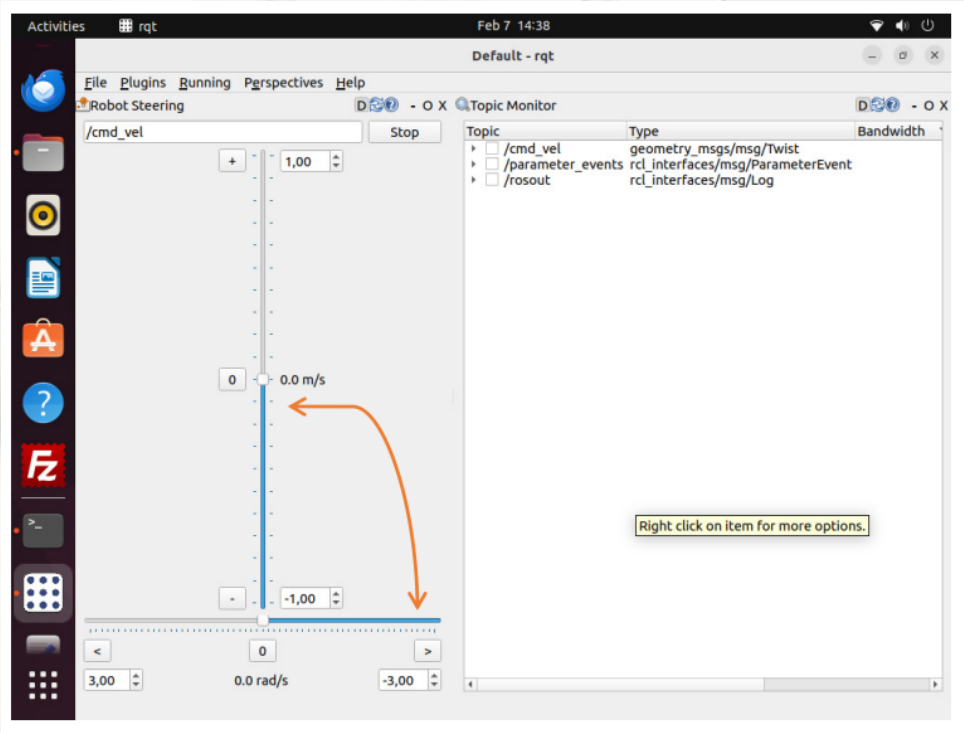
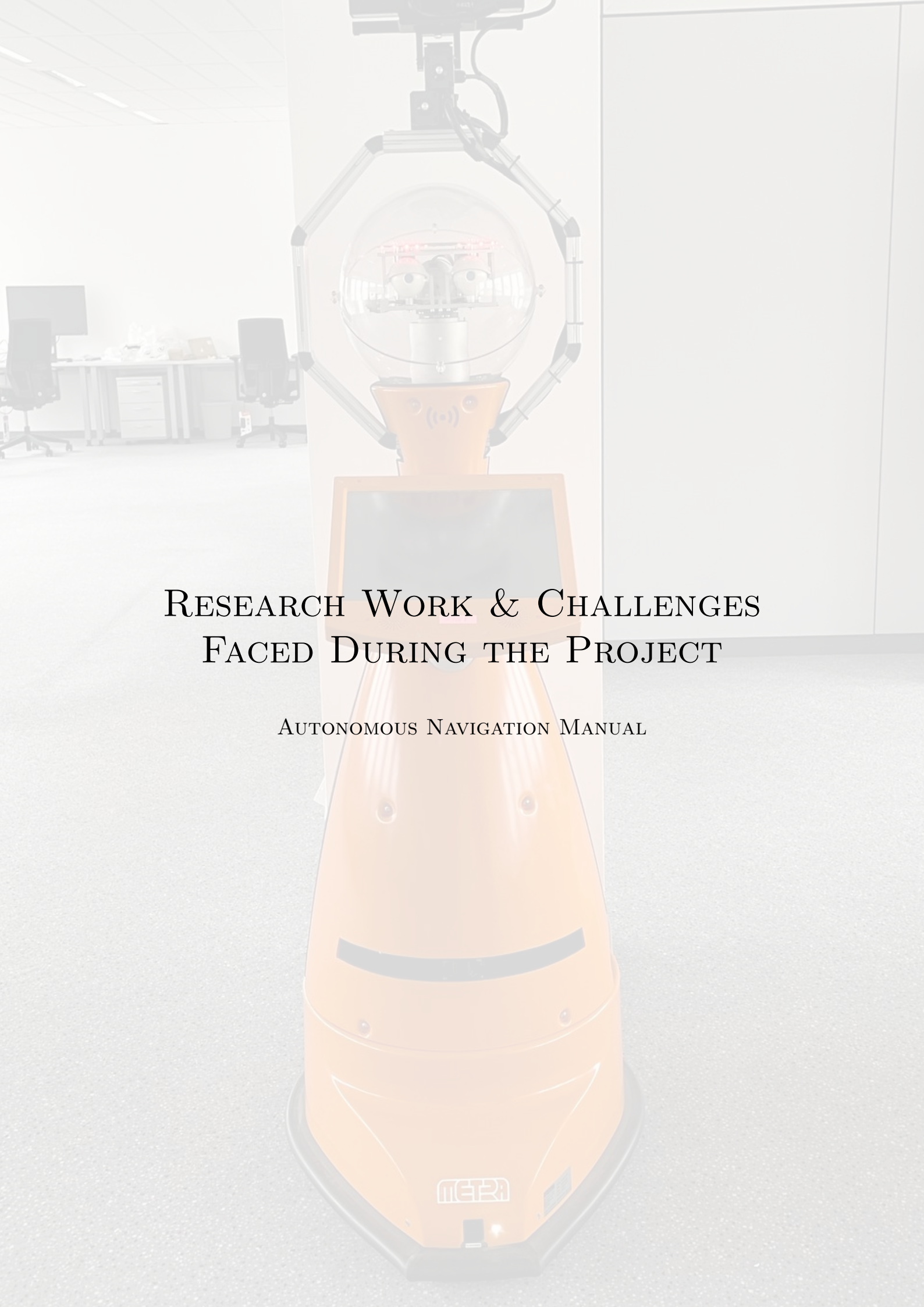


Figure 22: Use these sliders.

- For directional changes, use the horizontal slider. Sliding left initiates a leftward movement, right for a rightward direction, and keeping it centered drives the robot straight ahead.

8. **Closing the Tool**

- Once interaction with the robot's motion is completed, close(top righ corner) the tool in a standard manner to conclude the navigation session.



Research Work & Challenges Faced During the Project

Autonomous Navigation Manual

Integrating ROS and MIRA in the Metra SCITOS A5 Robot.

Yernar Kubegenov, Suraj Varma, Fikrat Mutallimov

Project Head: Prof. Dr.-Ing. Pascal Meißner
[Technical University of Applied Sciences Würzburg-Schweinfurt]

9th February, 2024

Abstract

This paper describes the process of integrating the Robot Operating System 2 (ROS2) framework into an existing MIRA-based robot navigation system, aiming to enhance the robot's navigation capabilities while ensuring compatibility with the MIRA software. The integration involved setup, problem-solving, and collaborative efforts, with early challenges including ROS2 operational errors and difficulties in hardware component drivers. Through persistent teamwork and exploration of technical aspects like localization and mapping using ROS2 Nav2 stack, we successfully integrated ROS2 into the robot's navigation system. Utilizing tools like MIRACenter(GUI) and ROS2-specific software facilitated smooth operation. The successful integration not only improved the robot's capabilities but also set the stage for future advancements in navigation technology, with potential for further enhancements by incorporating additional functionalities such as different cameras, sensors, or hardware components.

1 Introduction

Robotic navigation systems have become increasingly vital across various sectors, from industrial automation to service robotics. Integrating different software frameworks allows us to capitalize on each platform's strengths for optimal performance and functionality. In this paper, we detail our efforts to integrate the Robot Operating System 2 (ROS2) framework into an existing robot navigation system based on Mobile Intelligent Robotic Assistant (MIRA). Our aim was to enhance the robot's navigation capabilities while ensuring compatibility with the current MIRA software. The integration of ROS2 into the MIRA framework presented both challenges and opportunities. By combining ROS2's robustness with MIRA's versatility, we strived to create a navigation system that offered improved performance, flexibility, and adaptability to diverse environments. This paper offers a comprehensive overview of our approach, including information gathering, engagement strategies, critical integration challenges, project planning and execution, completed tasks, and planned actions. Through this work, we aim to contribute to the advancement of robotic navigation systems and provide valuable insights for future research and development endeavors in the field. Our experiences and findings could serve as a valuable resource for researchers, engineers, and practitioners involved in robotics and automation.

2 Approach towards the solution

To address the challenges associated with integrating ROS2 with the MIRA Framework for SCITOS robot navigation, a systematic approach was devised. Initially, extensive research was conducted to understand the intricacies of both ROS2 and MIRA Framework, including their functionalities, compatibility requirements, and potential integration strategies. Following this, a detailed methodology was outlined, encompassing steps such as hardware and software setup, installation procedures, and configuration settings. Each phase of the integration process was meticulously planned, executed, and documented to ensure clarity and reprehensibility. Additionally, proactive measures were taken to anticipate and mitigate potential errors or discrepancies encountered during the implementation phase.

Throughout the project, effective communication and collaboration among team members were prioritized to streamline efforts and maximize efficiency. This comprehensive plan served as a road map for successfully integrating ROS2 with the MIRA Framework, ultimately enabling SCITOS robot navigation with enhanced functionalities and capabilities.

Despite encountering issues stemming from the outdated nature of the SCITOS robot and the corresponding MIRA software, we diligently pursued solutions. This involved identifying the official version of MIRA compatible with Ubuntu 22.04 and resolving additional dependencies. Despite these initial hurdles, our comprehensive methodology, coupled with proactive problem-solving strategies, ensured the successful integration of ROS2 with the MIRA Framework. This facilitated the enhancement of SCITOS robot navigation capabilities.

3 Hardware Information of the Robot

The robot's hardware setup includes a complex control system and various sensors designed for perceiving its environment and interacting with it. The control system is split into base logic and a standard computer, which efficiently manages motor functions, collects data from sensors (such as magnetic, laser, and Kinect sensors), and distributes power. Safety features are built-in to ensure the system shuts down automatically during collisions, preventing damage. The computer, powered by an Intel Core i7-6700T processor, facilitates communication and control via a CAN-Bus connection. Sensors include the Microsoft Kinect for RGB and depth perception, the Asus Xtion Pro Live with RGB, depth perception, and a microphone, front and rear lasers for distance measurement, an RFID reader for token detection, and bumpers with a magnet sensor for collision detection and tracking. Mounted on the robot's head, the Kinect is controlled by Flir motion control motors using the MIRA-Commercial package. Retrieving data from the Kinect is made possible through the OpenKinect project, which utilizes libFreenect2 on Linux systems, and further processing is done using OpenCV for image processing and machine vision tasks. Charging options include both direct rear-side plug charging and automated charging via contacts on the robot's underside with a separate station. Connectivity is provided through an external USB port, while user interaction is facilitated by a 14" touch display with integrated speakers, offering a resolution of 1024x768 pixels.

Figure 23: SCITOS A5 Robot

4 Software Details of the Robot

4.1 MIRA framework

MIRA is a versatile cross-platform framework crafted in C++, offering a middleware, essential functionalities, and a plethora of tools for crafting and testing distributed software modules. Its primary focus lies in streamlining the development of robotic applications, but its adaptable architecture allows for seamless intra- and inter-process data exchange, catering to a broader spectrum of applications beyond robotics. Developed through collaboration between MetraLabs GmbH and Ilmenau University of Technology, MIRA caters to the needs of both commercial and educational sectors. The framework prioritizes the creation of complex, dynamic applications while promoting the reuse of modules as plugins, ensuring flexibility and scalability[2]. MIRA features MIRACenter, a graphical user interface (GUI) that provides users with a centralized platform for managing and visualizing various components and functionalities within the MIRA framework.

4.2 Robot Operation System (ROS)

In the ROS framework, the initial step in creating a new robot model involves constructing a URDF representation. This model encompasses various aspects such as geometry, kinematics structure, joints, wheels, sensors, and actuators. MetraLabs provided specifications for the Scitos-G5 platform, which greatly aided in the integration process. Once enriched with simulation-specific attributes, the URDF model seamlessly transforms into a comprehensive Gazebo simulation model, incorporating all sensors and actuators. Subsequently, device drivers and controller ROS nodes are developed for each sensor and actuator component.

4.3 Communication between MIRA and ROS

We provided the code establishes a ROS2 LifecycleNode, 'ScitosMira' to manage the integration of SCITOS robot modules with the MIRA framework. It utilizes ROS2's intra-process communication for efficient message passing and employs the Boost library for string operations critical in processing module names. Upon configuration, the node reads a list of module names and a robot configuration in XML format from ROS parameters. These modules, representing specific SCITOS hardware functionalities, are then dynamically instantiated through a factory pattern, leveraging MIRA's remote procedure call (RPC) mechanism for module communication and control. The 'ScitosMira' node implements the ROS2 lifecycle management callbacks ('on_configure', 'on_activate', 'on_deactivate', 'on_cleanup', 'on_shutdown')to manage the node's state transitions, ensuring a clean initialization and shutdown process. During the configuration phase, the node initializes the MIRA framework, starts it after a brief delay, and creates and configures the specified SCITOS modules. Error handling is integral, with checks for parameter existence and module registration ensuring robust operation. Additionally, the 'ScitosModule' class encapsulates functionality for interacting with the SCITOS robot via MIRA services, including service calls and parameter setting/getting, demonstrating direct integration between ROS2 nodes and the underlying robot control framework. The main function initializes the ROS2 environment, creates the 'ScitosMira' node, and adds it to a single-threaded executor, highlighting the entry point for the node's operation within a ROS2 ecosystem.Also,the scitos_mira package offers drivers for the Metralabs Scitos robot base (HG4 version) using the MIRA framework, facilitating integration with ROS2. Authored by Alberto Tudela, which was inspired by Chris Burbridge[6], these drivers expose various sensors and components such as Charger for battery monitoring, Display for status control, Drive for motor management, and EBC for extra device power control. Tested on ROS2 Humble and Ubuntu 22.04, the package is a port of scitos_drivers to ROS2, though frequent updates are expected. Development of our MIRA-ROS bridge is based on Alberto Tudela's work[5].

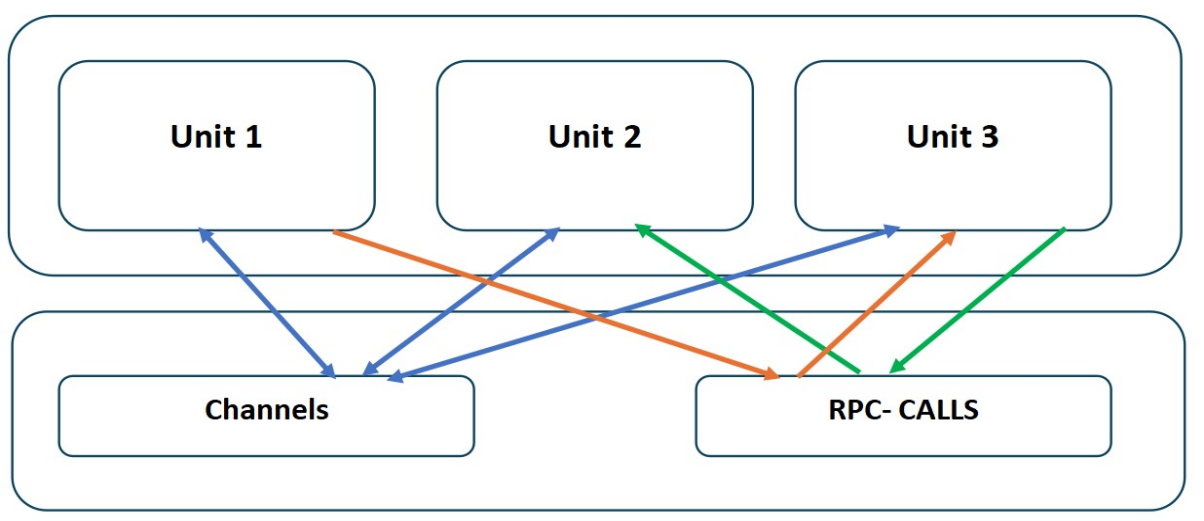


Figure 24: MIRA software architecture. The software system is built around two main parts: units and channels. Units are like small software building blocks that do specific tasks. They are connected together by channels, which are like virtual wires that let them talk to each other. Each unit can send and receive information through these channels or make new channels if needed. There are two types of units: ones that do things at set times and others that wait for new information before doing something.

5 Challenges Encountered in the Development Process

5.1 Installation and Update Challenges

The challenges encountered during the installation and configuration of the MIRA framework on Ubuntu 22.04 proved to be multifaceted. Initially, difficulties arose due to the outdated binary installer and the absence of a version tailored specifically for Ubuntu 22.04. This necessitated attempts to install MIRA from both source code and scripts, both of which failed to yield the desired results. Moreover, the process was compounded by additional dependencies and missing packages, which further hindered progress.

5.2 Developing and Troubleshooting

Despite efforts to update the MIRA framework and Scitos packages, issues persisted, including segmentation faults and missing drivers. The segmentation fault encountered during the execution of SCITOSDriver.xml pointed towards unresolved dependencies or compatibility issues between the MIRA framework and the underlying hardware components. Additionally, errors related to the absence of /dev/pcan32 highlighted potential communication problems, indicating a need for comprehensive troubleshooting of the CAN connection. The process of resolving these challenges involved a series of experiments, including rebuilding the MetraLabs-public packages, recompiling source code, and configuring environment variables. These experiments aimed to address compilation errors, missing files, and misconfigurations that hindered the proper functioning of the MIRA framework and Scitos packages. However, despite these efforts, certain issues persisted, necessitating further investigation and iterative refinement of the installation process.

5.3 Collaborative Solutions and Guidance

Experiments also involved seeking guidance from technical support and consulting with experts to decipher error logs, identify missing dependencies, and troubleshoot hardware-specific issues. Collaboration with the support team provided valuable insights into CAN communication, device configurations, and recommended installation procedures, guiding the team towards potential solutions and workarounds for encountered challenges.

6 Lesson Learned

6.1 Thorough Debugging and Troubleshooting

Our experience with integrating ROS 2 and MIRA highlighted the critical importance of thorough debugging and troubleshooting. We encountered various errors and challenges throughout the process, ranging from segmentation faults to missing drivers and configuration issues. By meticulously examining error logs and iterative experiments with different solutions, we were able to identify root causes and implement effective resolutions. This experience underscored the value of persistence and systematic problem-solving in overcoming complex integration issues.

6.2 Consideration of Hardware and Software Compatibility

Another key lesson learned was the importance of considering hardware and software compatibility when integrating different frameworks. Our project involved updating the MIRA framework, which posed compatibility challenges. We learned the significance of ensuring alignment between hardware specifications, software dependencies, and framework versions to avoid compatibility issues. This included verifying hardware drivers, updating dependencies, and carefully configuring environment variables to facilitate smooth integration. Going forward, we recognize the need for meticulous planning and assessment of compatibility requirements to streamline integration processes and enhance system stability.

6.3 Optimization for Performance and Efficiency

Additionally, our integration journey emphasized the importance of optimizing performance and efficiency in robotics projects. We encountered performance bottlenecks, particularly during compilation and execution phases, which impacted project timelines. Through iterative refinement of compilation processes, proper utilization of resources, and optimization of codebase, we were able to enhance system performance and efficiency. This experience highlighted the significance of prioritizing performance optimization strategies, to minimize project delays and maximize overall system performance. Moving forward, we aim to incorporate performance optimization as a fundamental aspect of our development workflow to ensure the successful execution of future robotics projects.

7 Conclusion

In our paper, we recount the challenging yet rewarding process of integrating the SCITOS Robot with both the MIRA Framework and ROS2. Despite initial hurdles in configuring ROS2 nodes and compiling necessary files, our team persisted through systematic troubleshooting and collaboration with external sources like MetraLabs, a company specializing in robotics solutions, and STRANDS, a project aimed at producing intelligent mobile robots capable of operating for extended periods in dynamic human environments. Through meticulous efforts, including software reinstallation and detailed error documentation, we successfully achieved our goal of integrating ROS2 alongside MIRA for robot navigation. By formatting the existing software, installing the latest Ubuntu and ROS2 versions, and configuring settings, we empowered the robot to perform navigation tasks previously reliant solely on MIRA. Our journey underscored the importance of adaptability, technical competence, and effective communication in complex technological endeavors.

8 References

1. *METRA, "MetraLabs mobile robots"*- viewed 5th May 2023. `https://www.metralabs.com/en/scitos-a5-2/`
2. *Ubuntu V 22.04.3 LTS*- viewed 29th May 2023. `https://ubuntu.com/`
3. *MIRA Framework Documentation*- viewed 14th October 2023. `https://www.mira-project.org/joomla-mira/index.php/installation`
4. *ROS 2 Documentation*- viewed 12th December 2023. `https://docs.ros.org/en/humble/`

5. *Alberto Tudela. "scitos_mira"*- viewed 13th January 2023. https://github.com/grupo-avispa/scitos2/tree/humble/scitos_mira

6. *Chris Burbridge. "scitos_mira" (Strands Project).*- viewed 21th January 2023. https://github.com/strands-project/scitos_drivers/blob/indigo-devel/scitos_mira/