D7.2 - Smart wheelchair hardware platforms for data collection and software integration

Due date of deliverable: 31.03.2024

Responsible partner: DFKI

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement **No 101070028-2**

Document Details

Deliverable No.	D7.2
Work Package	WP7
Task	T7.2
Deliverable Type	DEM — Demonstrator, pilot, prototype
Lead Partner	DFKI
Contributing Partner(s)	—
Due date of deliverable	31.03.2024
Actual submission date	31.03.2024
Abstract	This document describes the retrofitting of an electrically powered wheelchair with sensors and a compute unit, resulting in a hardware platform for smart wheelchair research.
Keywords	smart wheelchair, retrofitting, hardware equipment, LiDAR, wheel encoder, odometer, RGBD-camera, CAN-bus, Edge AI device

Dissemination level:

PU	Public	X
RE	Restricted to a group specified by the consortium (including Commission Services)	
CO	Confidential, only for members of the consortium (including Commission Services)	

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement **No 101070028-2**

Change History

Version	Date	Changed by	Changes Made
1	12.12.2023	Serge Autexier	First draft of table of contents
2	15.02.2024	Jan Janssen, Christian Mandel	Initial version of sections' content
3	08.03.2024	Jan Janssen, Zeming Duan, Christian Mandel, Serge Autexier	Content completion for internal review
4	20.03.2024	Jan Janssen, Christian Mandel, Serge Autexier	Adressing comments from internal review

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement **No 101070028-2**

Table of Contents

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement **No 101070028-2**

List of Figures

List of Tables

Acronyms Table

Acronym	Expanded Form
AI	Artificial Intelligence
CAN	Controller Area Network
CI	Continuous Integration
DDS	Data Distribution Service
DNN-LNA	Deep-Neural-Network Local Navigation Approach
DHCP	Dynamic Host Configuration Protocol
LiDAR	Light Detection and Ranging
PCL	Point Cloud Library
REXASI-PRO	REliable & eXplainable Swarm Intelligence for People with Reduced mObility
RGB	A camera with a normal RGB (Red Green and Blue) image
RGBD	A camera with a normal RGB image and an additional depth channel
ROS2	Robot Operating System 2

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement **No 101070028-2**

1. Overview of the deliverable

This deliverable describes the setup of the wheelchairs, in regards to hardware and software, used in the REXASI-PRO project.

1.1. Scope

This report describes the retrofitting of a commercially available electrically powered wheelchair with sensorial equipment and a compute unit. The resulting *smart wheelchair platform* is ready to integrate the different algorithms developed in REXASI-PRO, dedicated to implement safe and socially adequate driving behavior. Particular attention is paid to the properties of the additionally installed sensors. For example, the described resolution of the exteroceptive sensors should provide an idea of the robotic capabilities that can be achieved. This report also describes a basic software framework for the wheelchair, allowing to interface the different devices and providing basic sensor usage.

1.2. Audience

The deliverable described here is classified as public with regard to its dissemination level and is intended for the following audience: i) consortium members, ii) European Commission and iii) any other person interested in the topic.

1.3. Summary

This document describes the setup of the *smart wheelchair platform* used in the REXASI-PRO project, based on a commercially available electric wheelchair. In addition to the presentation of the hardware and sensors used, the ROS2 software framework used is also discussed (workspaces, scripts, messages, nodes, etc.).

1.4. Structure

The remainder of this document is structured as follows: In section 2 we introduce into the state of the art of smart wheelchair setup. Section 3 continuous with a description of the actual hardware added to the wheelchair, before section 4 details the basic software framework. In section 5 we finally conclude with a summary of this task.

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement **No 101070028-2**

2. Introduction

Since many years, researchers retrofit electrically powered wheelchairs by sensors and compute units in order to support people in steering-, navigation-, and obstacle avoidance-tasks. So-called *smart wheelchairs* [6, 11] can not only compensate for drivers' missing capabilities but also support in the training of powered wheelchair usage, for example w.r.t. spatial cognition. In the spirit of these developments deliverable D7.2 describes the base hardware and software integration of the smart wheelchair used in the REXASI-PRO project.

Such an ambitious goal cannot be achieved without the use of established and high-performance sub-components. Based on a robust wheelchair model from a German manufacturer with many years of development and production experience (i.e. *Otto Bock Mobility Solutions*), and with the help of sensors for environmental perception, some of which meet industrial standards, a smart wheelchair prototype was built for the REXASI-PRO project. But the software framework also had to meet high standards. Low-latency interprocess communication, as well as the availability of a large selection of existing robotics and 2D/3D computer graphics libraries were central criteria in the selection. Ultimately, the choice for the project fell on the *Robot Operating System 2* (ROS2) framework [9, 10], operated under the *Ubuntu 22.04* Linux OS. Beside ROS2, free libraries such as the *Point Cloud Library* (PCL) [14], *OpenCV* [3], *TensorFlow* [1] and others are to be used in REXASI-PRO.

From both the hardware and the software side, the design concept pursued here can be summarized under the keyword modularity. The installed laser scanners and depth image cameras are in principle interchangeable. This means that if a single sensor needs to be replaced, all that needs to be done is to reconfigure the mounting pose in the software framework and run the corresponding driver in the startup scripts. This approach makes the system built extremely future-proof. The ROS2-based software framework can also be described as modular. When the system is running, we can start new process nodes or terminate those no longer needed. Startup scripts are primarily written in the Python language. This means that here too, global scripts can easily import sub-scripts and thus be easily adapted to new configuration scenarios.

The selection of exteroceptive sensors should also be discussed at this point. We tried to provide a complementary combination with the 2D laser scanners and the 3D depth image cameras. While the forward and rear-looking LiDAR sensors with their large opening angles and high measurement precision form the basis for environmental detection, the two forward-looking RGBD cameras provide an additional option for obstacle detection. However, these sensors require more computing power (3D vs. 2D processing chain), cover a smaller viewing angle, and offer significantly worse measurement noise compared to LiDAR sensors.

The remaining hardware components also determine the quality of the overall system. The most important thing here is the selection of the computer. For this project, we chose an ARM®-based system with an integrated NVIDIA Ampere GPU instead of an x86-64 CPU/GPU combination. Particularly in view of the limited energy available (wheelchair batteries), this choice offers an optimal ratio of energy consumption to computing performance.

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement **No 101070028-2**

3. Hardware Setup

3.1. Basic Wheelchair Model

The base wheelchair model is given by the *c1000ds*[13], manufactured by the German company *Otto Bock*. This selection was primarily made because of prior experiences with the Otto Bock wheelchair controller used in this model, the *Curtis enAble50*. The wheelchair itself is equipped with two differentially driven front-wheels and two castor-wheels in the back. This non-holonomic kinematic design allows the wheelchair to rotate around the center of the front axle while standing still, and provides smooth steering while in motion. The wheels are maintenance free solid tyres, that are not filled with air. In order to mitigate smaller surface irregularities, the wheelchair has multiple spring-loaded joints.

Figure 1: *Front-, side-, and rear-view of the unmodified Otto Bock c1000ds wheelchair.*

The following sections detail the different hardware components integrated on the wheelchair. An overview of the complete system is given in section 3.11.

3.2. Fusebox

A fusebox that has slots for 12 different fuses was added to the wheelchair to protect the attached circuits from overloading and also to have a simple way to toggle a specific circuit in case of a problem. This fusebox is attached to a single power button which is directly connected to both batteries, resulting in a $24\,\mathrm{V}$ circuit.

3.3. CAN-Bus Interface

The CAN-Bus is used to gain access to the wheelchair controller, this enables us to override the commandeered joystick commands, control the lights and blinker as well as be able to read the current charging level of the wheelchair. Due to our prior experiences with the curtis enAble50 controller we already knew many of the CAN-Packages and the workflow that was needed to change the steering. This enabled quick tests with multiple USB CAN adapters to verify the packages did not change. Due to issues with linux support we did some initial testing on a

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement **No 101070028-2**

Windows based computer as a proof of concept, because the hardware and software was available already. Since we did not yet have a compute unit selected we did many tests with different linux enabled CAN-Adapters and Arduinos connected to CAN-Transceivers like the MCP2515[12]. The initial tests with a candleLight CAN adapter[8] had mixed results as the stick was able to receive packages just fine, but as soon as a package was sent the system was stuck in a loop to wait for acknowledgement of the package. Afterwards some tests were made with an arduino uno and an MCP2515 based transceiver board. After some initial successful tests a firmware was written that uses the same protocol as our electronic box used on the old wheelchairs. This enabled simple development with a laptop. With the delivery of the compute unit another node was created to use the CAN-Interface of the NVIDIA Jetson compute unit, this enables the lowest possible latency to receive and transmit packages, whereas with the arduino solution there was always another node introduced into the chain that could potentially cause slowdowns or even break.

3.4. Odometry

One of the base problems for wheeled mobile robots, in our case for the smart wheelchair used in REXASI-PRO, is to provide precise velocity measurements for the actuated wheels. With this information one can give a first estimate of the vehicle's driven trajectory and herewith of its pose within a map [2]. This process is called *dead reckoning* or *odometry*. In order to setup a robust and precise odometry system for the wheelchair (cf. Fig. 2), we installed a 120 teeth C45 steel gear on the left and right drive axle. Additionally a *GEL248V* incremental encoder [7] has been installed very close to the gear. This combination provides 480 encoder ticks per complete wheel rotation, resulting in detectable rotation offsets of approx. $2.5\,\mathrm{mm}$ on the ground or $0.75°$. The incremental encoder are directly powered by the $24\,\mathrm{V}$ fusebox.

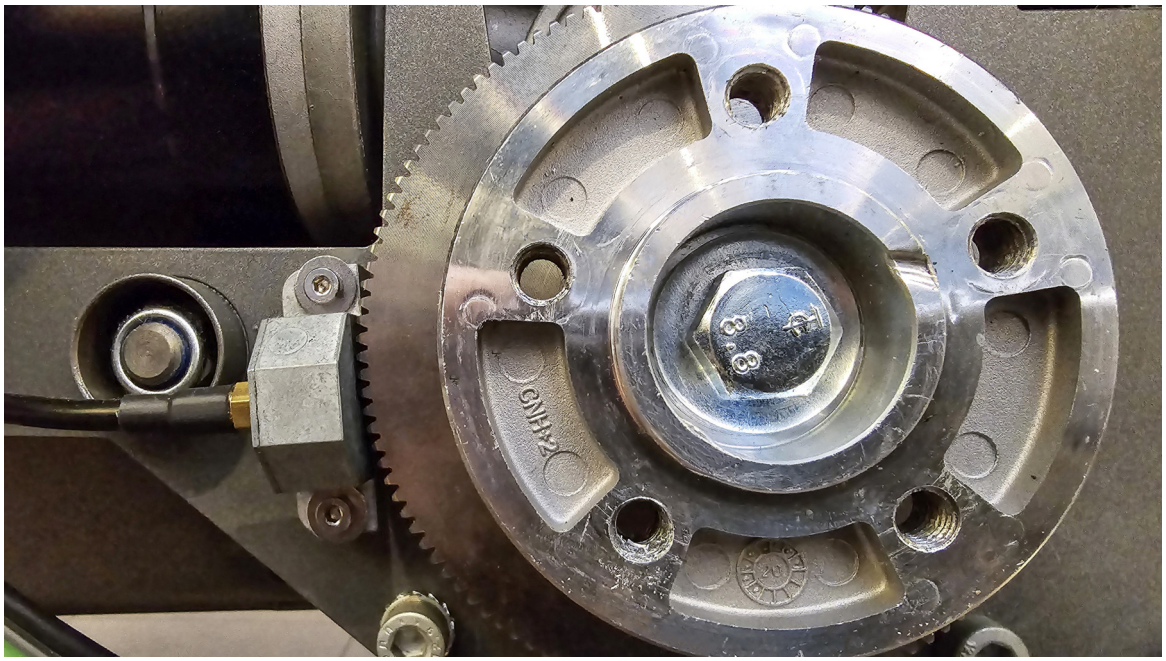

Figure 2: *Wheel encoder mounted on the axle of the front-right actuated wheel.*

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement **No 101070028-2**

3.5. Compute Unit

The wheelchair's main compute unit is a *NVIDIA Jetson Orin™ NX 16GB*-based *reComputer Industrial J4012 Edge AI* device [15] (cf. Fig. 3). Its main features are given by an 8-core ARM® Cortex-A78AE CPU, a 1024-core NVIDIA Ampere GPU with 32 Tensor-cores, 16GB of LPDDR5 RAM, two RJ45 GbE ports, as well as its robust heatsink aluminum case. We equipped the compute unit with a *Western Digital BLACK SN850X* M2 SSD, featuring 4 TB of capacity, write speeds of up to 6600 MB/s and read speeds of up to 7300 MB/s. The J4012 can be operated fanless. But since this introduces thermal throttling during longer usage, e.g. during training data recording, we decided to mount an additional 120 mm case fan, which solves this problem. The computer is mounted on the rear rack of the wheelchair and is connected to the 24 V fusebox via an additional 9 V-32 V to 24 V voltage converter. This setup provides stable and safe power supply to the unit.

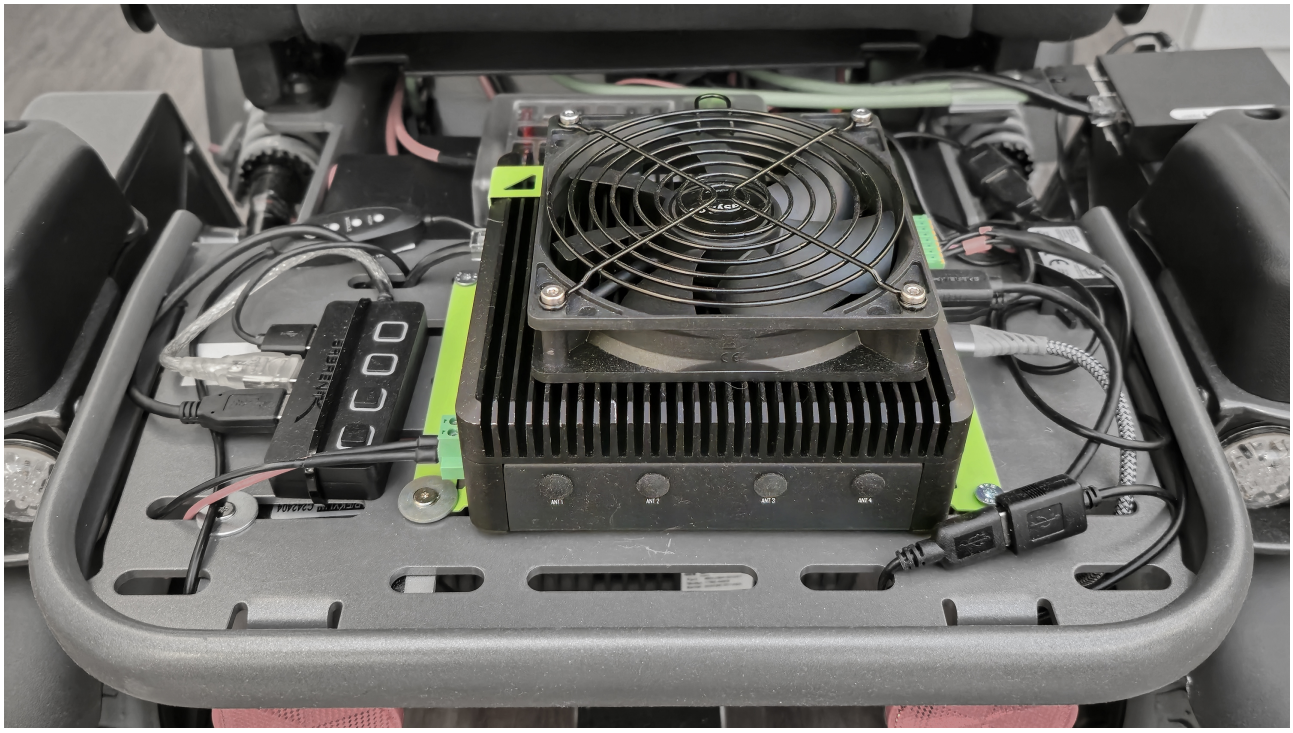

Figure 3: *NVIDIA Jetson Orin™ NX 16GB-based compute unit mounted to the rear rack of the wheelchair. Note the case fan mounted on top of the compute unit, allowing for unthrottled operation under high load.*

3.6. Laser Scanners

Laser scanners, or light detection and ranging (LiDAR) sensors, are used to measure distances to nearby obstacles. They do so by measuring the transit time of focussed infrared light beams from the sender to the obstacle and back to a receiver. Laser scanners usually scan in one or more two-dimensional planes that *slice* the environment. For the REXASI-PRO wheelchair we selected two *SICK TiM 571-2050101* laser scanners [16] attached to the front and to the back of the wheelchair on a dedicated mounting plate 10 cm above ground (cf. Fig. 4). The laser scanners are connected to a network switch locally mounted on the wheelchair, and are powered over a fuse box connected to the 24 V board power. Both scanners provide scans at a frequency of 15 Hz and can scan a two-dimensional area

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement **No 101070028-2**

of max $270°$ with an angular resolution of $0.33°$ and a mean distance measurement error of $<20\,\text{mm}$ given $90\,\%$ remission. Due to mounting restrictions, we had to crop the opening angles of both laser scanners to $\pm 79°$. Technically it would have been possible to extend the mounting bracket such that a wider field of view would have been achieved. The major reason for not doing so was that we would have had to change the wheelchair's outer shape, rendering it less maneuverable in narrow environments. Additionally this could have let the mounting bracket vibrate up and down, introducing noise to the measurements. Ultimately the mounting bracket has been chosen to be as short as feasible in order to avoid these problems. The bracket is made out of a $5\,\text{mm}$ stainless steel that is bend on both sides. It can withstand vehicle's vibrations without beginning to vibrate itself.

Figure 4: *From left to right: front laser scanner, rear laser scanner, and CAD-rendering of LiDAR-mount designed by DFKI. Both laserscanners are protected by cages that allow for accurate alignment of the sensor.*

3.7. RGBD Cameras

In addition to the LiDAR sensors, two RGBD cameras were mounted on the wheelchair. This kind of camera provides usual RGB image streams and additionally depth image streams in which each pixel describes the distance to the sensed object. RGB images and depth images are calibrated to each other, such that depth and color images can be overlayed, e.g. for rendering colorized pointclouds [5]. Both cameras belong to the *Intel® RealSense™* series of sensor devices. The camera horizontally mounted at the end of the right armrest is a *D455*, while the vertically mounted and slightly downward pitched camera is a *D415* (cf. Fig. 5). Both cameras are primarily used for obstacle detection. This includes the analysis of sensed depth streams and the derived pointclouds. Here we generate additional 2D occupancy grid layers by slicing the pointcloud and by more sophisticated approaches that search for impassable holes in the ground plane, e.g. downward stairs. Additionally, the RGB image stream is used for people detection.

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement **No 101070028-2**

Figure 5: *Vertically and horizontally RGBD-cameras mounted on the left and right armrest respectively.*

3.8. Streamdeck

As an easy to use human machine interface device we selected the *Elgato Stream Deck mini* [4] (cf. Fig. 6). The device provides 6 physical buttons that are able to display animated or still images. These buttons are used to inform about the state of the wheelchair's various subsystems, including current RGB images, an occupancy map depicting the integrated percepts of the laser scanners, the current CPU/GPU usage, power drain of the compute unit, battery level and more. With the button functionalities, the user can (re-)start and stop processes, or shutdown the whole system. The Stream Deck is mounted on the left wheelchair armrest in front of the emergency stop button and one of the two RGBD cameras.

Figure 6: *Stream Deck*

3.9. Emergency stop

An emergency stop button was also attached to the wheelchair, this can stop the wheelchair safely if the system misbehaves. It is a secondary option to shut down the system; if possible the regular power button on the joystick should be used in those cases, because it safely shuts down the wheelchair controller, and the software on the wheelchair can be notified of this.

3.10. Network Architecture

To connect the laser scanners to the compute unit and to enable remote development a simple network switch was mounted to the wheelchair, see Fig. 7. All devices in the network need to have a static IP-Address since there is currently no DHCP-Server on the wheelchair.

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement **No 101070028-2**

Figure 7: *Network switch*

3.11. Complete system

The hardware components described in sections 3.2 to 3.10 were integrated into the wheelchair system (c.f. section 3.1). The prototype constructed in this way can be seen in Fig. 8; the positions of the sensors are highlighted in red. Please note that the information sign mounted on the headrest is only used as part of the training data collection. It draws people's attention to their rights regarding the general data protection regulation.

Figure 8: *Front-, side-, and rear-view of the retrofitted Otto Bock c1000ds wheelchair. Note that the sign mounted to the backrest is only used for training data collection sessions (cf.* Deliverable D7.3*), and informs about GDPR-related issues.*

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement **No 101070028-2**

4. Software Setup

The central infrastructure for REXASI-PRO software development at DFKI consists of a *GitLab* instance hosted at the DFKI in Bremen. This revision system is used not only by the DFKI, but also by project partners who need access to the repositories for the corresponding work packages.

The basis for the developed software stack is provided by the *Robot Operating System - ROS2 Humble* [9]. This version is part of the *Ubuntu 22.04* operating system version, which itself was selected as the default Linux operating system for the REXASI-PRO project. ROS2 was chosen because it represents the de facto standard in royalty-free robotics software development. ROS2 also offers a large number of libraries that, for example, implement communication with sensors or implement standard algorithms in areas such as computer graphics, self-localization, mapping, and path planning.

4.1. Workspace

We created a Visual Studio Code (vscode) workspace, that can be either used in docker or natively. The workspace can also be used with any other editor by launching the include bash files for the given tasks. We selected vscode as our base editor because it is a very extendable but still low weight and open source. Additionally it can be used very easily remotely via the official *Remote - SSH* extension provided by Microsoft. This enables easy remote development directly on the target machine, for example the wheelchair. It also has very well supported multiplatform features, enabling usage on linux, windows or mac devices.

Scripts

There are multiple larger scripts used in the workspace to ease development, see Table 3. These can be either launched directly via bash or as a job from vscode.

CI Scripts

Additionally there are some scripts that are only needed for the CI (Continuous Integration) runners, see Table 4. These CI pipelines are used to verify the code in the repositories can be built and all files needed to function are available. Due to the complex nature of the ROS2 dataflow and execution, automatic unit tests are not included in the CI-Pipelines. But the code is tested on the production hardware directly to verify the complete stack is working without any issues.

4.2. Packages

Rolland

This is a meta package used to host all relevant configs, parameters and launch files, as well as the urdf and meshes of the wheelchair. It also contains a python package called *Rolland*[1] that contains all methods needed in the launch files. This enables easy reuse and simple modification of components and subsystems of the wheelchair.

[1]The name of this package is maintained for historical reasons and names the series of automatic wheelchairs build in the group since the end of the 1990s in a combination of *rolling* and *Roland*, which is the famous statue on the market square of Bremen.

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement **No 101070028-2**

Table 3: *Scripts designed to ease workspace development.*

Name	Description
ubuntu_22.installer.sh	This installer automates the installation of all needed packages and the setup of the udev rules needed for the connected microcontrollers.
setup.sh	This is used to setup the workspace after cloning just the workspace repository. It clones all the relevant repositories and will then install all required dependencies via rosdep.
pull.sh	This will clone new or pull existing relevant repositories and will then install all required dependencies via rosdep.
build.sh	Build the complete workspace. This will need to check everything in *src* for packages and if the package needs a rebuild due to changes to the source code or build files.
build.sh <package>	Build only the given package and its dependencies. This is especially useful in a large environment where a full build can take some time, because the build system needs to check every package for changes.
status.sh	Checks all subrepositories for changes that should be commited to the repositories.
new_package.sh	Create a new package that is derived from our template, will automatically create the repository and upload it to gitlab.
createDevConfig.sh	Create the *.vscode/devcontainer.json* configuration based upon the architecture of the current device.
convert_https.sh	This script converts all urls in *src/ros2.repos* to use https instead of ssh.
convert_ssh.sh	This script converts all urls in *src/ros2.repos* to use ssh instead of https.

Additionally it is the default target base folder for maps, bags and for the extracted dataset for the AI training data.

xenotronic_driver

The xenotronic_driver package contains everything to connect to the low level hardware on the wheelchair.

It retains its name from the old wheelchair controller.

JetsonCanProvider

JetsonCanProvider provides the direct connection via CAN to the wheelchair. It uses the CAN network connection via SocketCAN. The primary function of this Node is to reroute the joystick commands send from the control unit of the wheel-

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement **No 101070028-2**

Table 4: *Continuous Integration (CI) Scripts used by gitlab.*

Name	Description
ci_CreateVCS.sh	This updates all the repository urls in src/ros2.repos to use a modified https version with a group token, so that the dev environment can access all repositories in the gitlab group.
ci_CreateDockerImage.sh	The script creates multiple dockerfiles, it uses a Dockerfile as input and injects all commands from *ubuntu_22.installer.sh*, after some filtering for commands that are not needed, or possible in docker. By splitting it into multiple commands its also easier to cache the artifacts, which allows better reusability of docker layers.
ci_build.sh	This is a wrapper script that launches multiple sub scripts to build in the CI environment. Its successful execution helps to verify that the commited code compiles and does not break any other package. These tests cannot verify that the code is behaving correctly, just that they can be built.
ci_doc.sh	Pulls the documentation repository and launches the included docker compose, resulting in updated documentation and images created from the package descriptions, which then will be commited and pushed automatically to the documentation repository. This results in always up to date documentation in a single repository for ease of use.

chair, change the given data, and send the new data to the wheelchair as if it came from the control unit.

JoystickDriving

This node takes received data of the type *sensor_msgs::msg::Joy* from */joy* and converts it into a *geometry_msgs::msg::Twist* and publishes it on */cmdVel*, it is used to resend the same data back to the JetsonCanProvider as was received. We make use of that during phases where the wheelchair data should not be changed, but it needs to be available to the system - for example during training data recording or while in manual driving mode.

OdometryProvider

The OdometryProvider parses data sent to */odometerReadings* topic, publishes it convert to a position tracked on */odometry* and changes the TF tree accordingly.

XenotronicProvider

To interface with the xenotronic control board on the older wheelchair models the XenotronicProvider is used. This receives the data from the joystick and publishes it to */joy*, it also receives the ticks from the odometers and publishes that to */odometerReadings*. Additionally it can receive commands to be sent to the wheelchair on */cmdVel* as a *geometry_msgs::msg::Twist* and on */failSafeDriveRequest* as a *rolland_messages::msg::DriveRequestMsg*. Both topics can theoretically

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement **No 101070028-2**

be used in parallel but the newest command always overwrites the older commands sent to both topics.

SoloOdometryProvider

The SoloOdometryProvider is used to receive the odometerticks from the arduino mega that connects to the rotary encoders, see subsection 3.4. These will then be propagated to */odometerReadings*.

4.3. ROS2 Framework and ROS2 Node Architecture

ROS 2 is a powerful middleware designed for developing robot control software. It builds upon the foundation of its predecessor, ROS 1, with several enhancements to meet the needs of both academic and commercial projects. ROS2 operates as a distributed real-time system. Various components within a robot, such as sensors, motion controllers, detection algorithms, and navigation algorithms, are represented as nodes. These nodes communicate with each other in a decentralized manner using a middleware called Data Distribution Service (DDS).

Nodes

Nodes are the fundamental building blocks in ROS2. Each node represents a specific functionality or task within the system. Examples of nodes include sensor drivers, motion planners, perception algorithms, etc. Nodes primarily communicate by publishing and subscribing to topics via DDS.

Topics

Topics are the base layer of multinode communication in ROS2. A topic always has a distinct data type and enables nodes to either publish or receive data in a defined format. The communication with topics is asynchronous, meaning a node will not wait for data to be received. Due to the asynchronous nature of the communication also the subscribing node will not have to actively poll for new data, but as soon as new data is received on a topic that is subscribed an event will be launched, for example a handler for the message.

Services

Services provide a request-response mechanism. A node can offer a service, and other nodes can call that service to request specific actions. Unlike topics, services are synchronous, meaning the caller has to wait for a response from the service provider.

Parameters

Parameters allow nodes to have tunable parameters that can be set during runtime. Nodes can read and write parameters, additionally they will be informed if a parameter changes, making it easy to adjust behavior without modifying code.

Actions

Actions extend the capabilities of services and allow for complex interactions, such as goal-setting, feedback, and cancellation.

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement **No 101070028-2**

An example of an action server might be a long-running task like navigation, with the ability to inform clients about the progress of the task.

Communication

ROS2 uses DDS as its communication middleware, this enables a decentralized publish-subscribe architecture, ensuring efficient and reliable data exchange between nodes. If data is only transmitted on a local machine the DDS architecture even enables to share the memory of a message directly without the need to pack and unpack a network transmission.

Simplified example of dataflow for obstacle mapping

A simplified example of the complex data flow used for obstacle mapping can be seen in Fig. 9. This shows the dataflow between the nodes in the current version, which will still change until the finalization of the 3D-safety-layer which will be shown in detail in the deliverable *D4.8*.

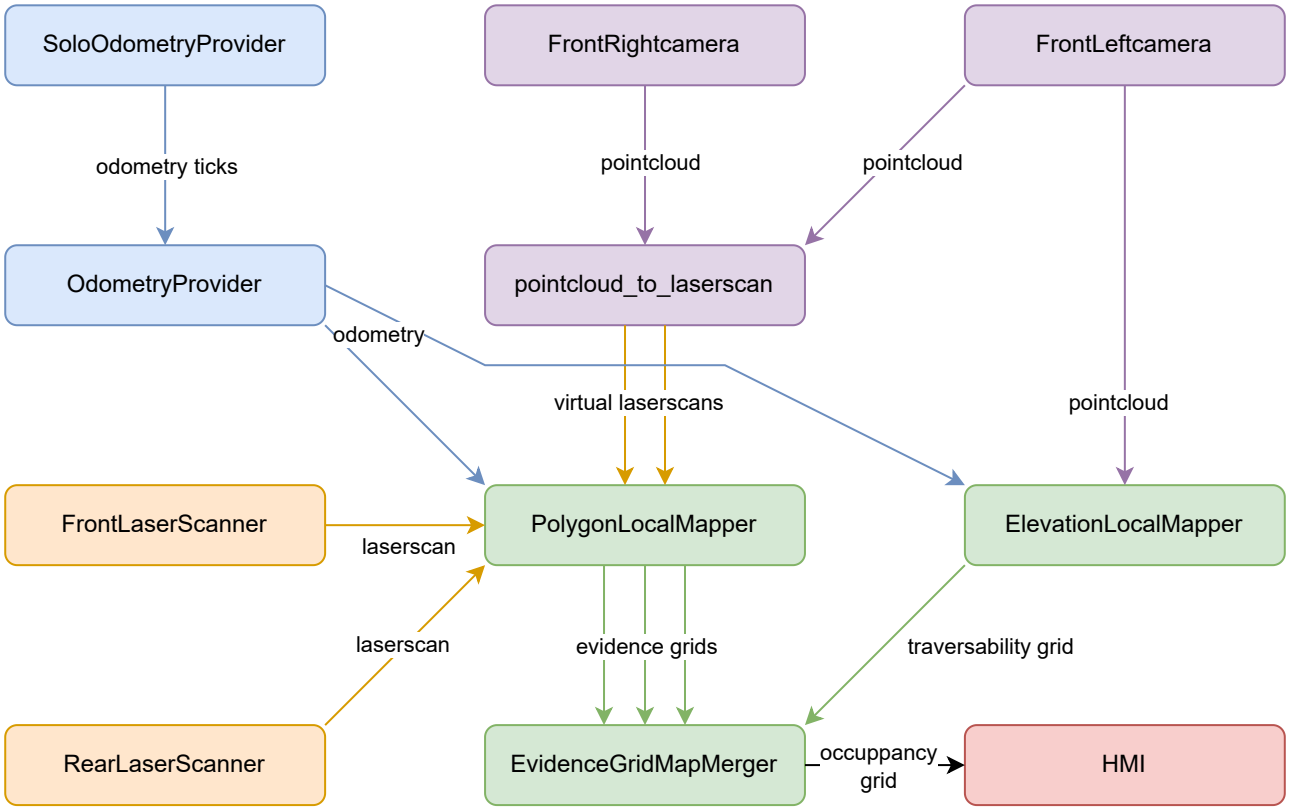

Figure 9: *Flow of data between nodes for the purpose of mapping local obstacles into an evidence grid.*

Integration with other components

The purpose of this deliverable is only to give a broad overview around the hardware and software framework that was created to enable development of the complete system. The final integration and full control architecture will be published within *D7.4*.

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement **No 101070028-2**

5. Conclusions

Smart wheelchairs are not yet available on the market in terms of comprehensive equipment with sensors, computing units and human-machine interfaces. This is mainly due to two reasons. On the one hand, the demand for wheelchairs fully equipped with robotic components is rather low, meaning that providers cannot place them on the market at reasonable prices. On the other hand, users generally do not want to overcompensate for their remaining skills with the help of high-tech, so that they atrophy through disuse.

In this area of tension, this document describes the incremental development of a smart wheelchair, starting with the retrofitting of an electrically powered wheelchair available on the market with sensors and a computing unit. This integrated hardware platform is operated using a free operating system (Ubuntu 22.04) and a free robotics software framework (ROS2). This REXASI-PRO wheelchair is not only used for the development and evaluation of individual project components, such as the Deep Neural Net-based Local Navigation Approach (DNN-LNA), but its modular construction also shows possibilities for gradually equipping with robotic components.

A central design parameter of the entire system is the performance of the selected computer. It should be mentioned that a final assessment of this size can only be made at a later point in time of the project. Only with the final determination of the network architecture of the DNN-LNA algorithm, as well as the potential integration of further neural networks, e.g. for the human-machine interface or people tracking, will it become clear whether there is enough computing power available. In the event that additional computing capacity is required, it is planned to add one or two NVIDIA Jetson Nano boards for the HMI or the people tracker. This will be reported in the deliverable *D7.4 Fully automatically driving smart wheelchairs for safe and smooth autonomous operation*.

Looking at this deliverable purely from a project perspective, it offers partners access to a comprehensive development platform. For example, the selected software infrastructure will be used at a later date to add additional components, such as a verbal human-machine interface, people tracking, and the interface to the orchestrator. In particular, Dockerization and the ability to clearly define interprocess communication protocols, here in-between nodes via topics, will minimize the integration effort. But the foundation for successful project integration was also laid from a hardware perspective. For example, the sensorial equipment of the wheelchair with RGBD cameras and LiDAR sensors can complement the drone sensors from project partner Hovering Solutions in the context of environment mapping.

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement **No 101070028-2**

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] J. Borenstein, H. R. Everett, and L. Feng. Where am i? sensors and methods for mobile robot positioning. Technical report, University of Michigan, 1996.

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[4] elgato. stream deck mini. `https://www.elgato.com/us/en/p/stream-deck-mini`, 2024. Accessed: 2024-02-13.

[5] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In *International Symposium on Experimental Robotics*, 2010.

[6] Jesse Leaman and Hung La. A comprehensive review of smart wheelchairs: Past, present and future. *IEEE Transactions on Human-Machine Systems*, 47(4):486–499, 2017.

[7] Lenord+Bauer. 2-channel speed sensor - Compact sensor with HTL or TTL output signals. `https://www.lenord.de/fileadmin/user_upload/dokumentation/ti_248_e.pdf`, 2023. Accessed: 2024-02-15.

[8] Linux Automation GMBH. candleLight USB-CAN-Interface. `https://linux-automation.com/media/pages/products/candlelight/Datasheet_candlelight.pdf`, 2023. Accessed: 2024-02-15.

[9] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074, 2022.

[10] Steven Macenski, Alberto Soragna, Michael Carroll, and Zhenpeng Ge. Impact of ros 2 node composition in robotic systems. *IEEE Robotics and Autonomous Letters (RA-L)*, 2023.

[11] Christian Mandel, Tim Laue, and Serge Autexier. Smart-wheelchairs. In Pablo Diez, editor, *Smart Wheelchairs and Brain-computer Interfaces*, pages 291–317. Elsevier; Amsterdam, 2018.

[12] Microchip. MCP2515 - Stand-Alone CAN Controller with SPI Interface. `https://ww1.microchip.com/downloads/en/DeviceDoc/MCP2515-Stand-Alone-CAN-Controller-with-SPI-20001801J.pdf`, 2019. Accessed: 2024-02-15.

[13] Ottobock. C1000 DS. `https://www.ottobock.de/rollstuehle/elektrorollstuehle/c1000ds/index.html`, 2021. Accessed: 2024-02-15.

[14] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE International Conference on Robotics and Automation*, pages 1–4, 2011.

[15] seed studio. reComputer Industrial J4012- Fanless Edge AI Device with Jetson Orin™ NX 16GB module. `https://www.seeedstudio.com/reComputer-Industrial-J4012-p-5684.html`, 2024. Accessed: 2024-02-13.

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement **No 101070028-2**

[16] Sick AG. TiM571-2050101. https://cdn.sick.com/media/pdf/4/44/444/dataSheet_TiM571-2050101_1075091_en.pdf, 2024. Accessed: 2024-02-15.

This project has received funding from the European Union's Horizon Europe research and innovation program under grant agreement **No 101070028-2**