A Framework for Developing and Evaluating Modular Mobility Aids for People with Visual Impairment: An Indoor Navigation Use Case

Florian von Zabiensky, Grigory Fridman, Sebastian Reuter, Oguz Özdemir, Michael Kreutzer, Diethelm Bienhaus

Institute of Technology and Computer Science
Technische Hochschule Mittelhessen
University of Applied Sciences
Gießen, Germany
{florian.von.zabiensky, grigory.fridman, oguz.oezdemir, sebastian.reuter, michael.kreutzer, diethelm.bienhaus}@mni.thm.de

Abstract—Electronic Travel Aids are devices that help people with visual impairments navigate and orient themselves. The development of such devices is often associated with a loss of time in repetitive work, resulting in slow progress in this field. A collaborative community that shares its expertise could accelerate this progress and lead to truly useful and market-ready products for visually impaired people. To make such an exchange efficient, a standardized, component-based ecosystem is required. So far, such an approach for Electronic Travel Aids has not been pursued in the literature and is therefore addressed in this paper. To this end, a model for identifying component boundaries is presented and illustrated by a project in the form of an ultra-wideband indoor navigation system. The advantages of such a component-based development in general are described. In particular, the use of the Robot Operating System 2 (ROS 2) for the implementation is highlighted and its suitability for such an ecosystem is discussed based on practical experience with it. The evaluation of such an ETA with the use of the ROS 2 ecosystem and a component-based ETA are also highlighted. The contribution of this work is a framework that reduces the effort for the development and evaluation of electronic travel aids and allows an early involvement of users in the development process.

Index Terms—ETA; electronic travel aid; mobility aid; ROS 2; ROS; robot operating system; component-based development; user-centered design

I. INTRODUCTION

This paper was originally presented at the SMART AC-CESSABILITY 2023 conference with a focus on developing electronic travel aids (ETAs) [1]. For the journal article it was extended by a user evaluation of the described ETA to show the possibilities of the presented framework beyond the development and the advantages it offers for user-centered development.

In the field of ETA an active research takes place. ETAs are devices that help people who are visual impaired or blind in travelling tasks like orientation, navigation, and obstacle avoidance. Several overview papers summarize relevant and representative research [2]–[5]. It is important to continually expand the possibilities of these aids and thus increase the mobility of those people.

However, when looking at the systematic literature review of Khan et al., one can find tables containing groups of papers classified by the technology or hardware components used [2]. If the projects behind those papers rely on the same technology or hardware, the efficiency of their development could be increased by sharing common source code or libraries. This would increase the time that can be spent, e.g., on designing user interfaces. For example, if one researches a novel approach to the acoustic representation of obstacles, he can focus on his research, knowing there is a pool of hardware components, simulation methods and algorithms for locating obstacles in the environment. In this way, all human resources can be concentrated on the novel representation to achieve faster and better results.

This example is only possible, if the development of ETAs is based on common principles. For this purpose, we divide ETAs into interchangeable components and propose a framework to develop ETA components that can be shared. Such a development is presented with an indoor navigation system as an example.

The experiences made during the development of this system, as well as the development itself, are finally discussed to openly present the positive and negative sides and to justify a recommendation for such a development.

This paper is structured as follows. First, it presents several literature reviews that summarize and classify numerous concrete projects from ETA research in section II. Section III discusses the problems identified in the research. As a solution proposal, section IV describes a specialized Human Machine Interface (HMI) for ETAs to divide ETAs into their components, as well as the Robot Operating System 2 (ROS 2) as a software development kit. To discuss this solution, section V applies it to an indoor navigation system as an example for development and evaluation. Section VI discusses the practical experience of the development, with its advantages and disadvantages. Section VII concludes the results and gives an outlook to further work needed to be done.

II. Related Work

With a focus on ETAs, there are several development projects in the field of navigation systems for blind and partially sighted people. These projects use different approaches to help these individuals navigate safely in both indoor and outdoor environments. With the digital transformation of healthcare, Internet of Things devices can enhance the capabilities that can be achieved in this area. Khan et al. [2] conducted a systematic literature review to analyse the challenges and opportunities of such 'smart navigation devices' that have been researched and developed over the last decade. Using structured selection criteria, the review identified 191 relevant articles published between 2011 and 2020 in six different peer-reviewed digital libraries.

Khan et al. [2] categorized various approaches to navigation systems for blind and visually impaired individuals into three parts. The study provides a comprehensive list of commonly used systems, tools, and hardware components as examples.

1) Approaches reported for navigation system development e.g.:
   - Indoor navigation system
   - Mobile application
   - Wearable navigation systems with e.g., smart watches
2) Technologies/tools proposed for navigation assistant development e.g.:
   - Raspberry Pi microcomputer
   - Android-based applications
   - Microcontroller
3) Hardware components proposed for obstacle avoidance e.g.:
   - Bluetooth beacons
   - Haptic devices
   - Ultrasonic sensors
   - Global Positioning System (GPS)

It is evident that various projects developed in these fields share similarities in terms of system level, technology, and hardware components used, indicating that multiple development efforts can result in similar or identical solutions. This indication is present in most of the recent survey papers in the field of ETAs [2]–[5]. A similar situation in the field of robotics was part of the driver for the Robot Operating System [8].

III. Problem Statement

In 2007, then PhD students Eric Berger and Keenan Wyrobek discovered a fundamental problem in robotics research. A pattern was emerging, in which researchers wanted to build on a proof-of-concept presented in a paper to implement their own idea. Either they lack details of the software used, or it is unusable for whatever reason, so they are often forced to spend 90 percent of their time rewriting other people's code and developing their own prototype test-bed. This leaves only the remaining 10 percent to develop their own innovation, which then lacks quality but enables the intended

publication. This creates a cycle of reinventing the wheel and wasting a huge amount of time. This led to the idea of creating a kind of Linux for robotics with the Robot Operating System (ROS), containing a common set of software and developer tools that would allow roboticists to build innovative ideas on the successes of others [9].

Looking at the numerous projects that have already emerged in the field of ETAs [2], one discovers this problem pattern again in many respects. In particular, the description of the selection process of the literature to be evaluated shows that many projects are similar and only a few add value to the state of the art. Additionally, these findings are rarely translated into products that benefit the end user. One reason for this might be that the projects are usually developed from scratch and thus valuable resources are lost to be put into the actual core of the work. For example, the categorization in [2] of some of the ETA prototypes known from research according to their hardware components makes it easy to see that many projects use similar, if not the same, subsystems and devices. The same concepts and technologies are being used for similar, if not identical, tasks. This leaves little time for iterative improvements and testing with visually impaired people. To counteract this, this paper presents a component-based development that contributes to the exchange between working groups and thus to a faster and more efficient prototype cycle. To this end, we built upon ROS 2, the successor of the above-mentioned ROS, which also serves as a motivator. In the field of mobile robotics, ROS 2 has helped components to be exchanged and to communicate with each other in a uniform manner, so that individual working groups can work much more efficiently on their research problems. In this paper, the development of an indoor navigation system using a vibration vest as an output device is presented. This project is not put in focus because there are other projects with similar results. The focus of this paper is on how the development can be made more efficient, and this will be shown and evaluated using the indoor navigation example.

IV. Solution Approach

If we look at the model of a human-machine interface in a very abstract way, it can be broken down according to Kantowitz and Sorkin [10] into the subcomponents shown in Fig. 1. A person (left) has the ability to acquire information through the senses available to him or her. This information is processed in the brain to make decisions based on it, such as operating the machine (right). The control components provided by the machine for this purpose have an influence on the internal state of the machine, from which outputs are generated to present information to the person. The two transitions between the human side and the machine side are called *human-machine interfaces*.

This model can also be used in an extended form to describe a visually impaired person and his mobility aid, where the physical environment is added as a crucial component (see Fig. 2). The ETA itself takes on the role of the machine by sensing relevant information about the environment (e.g.,
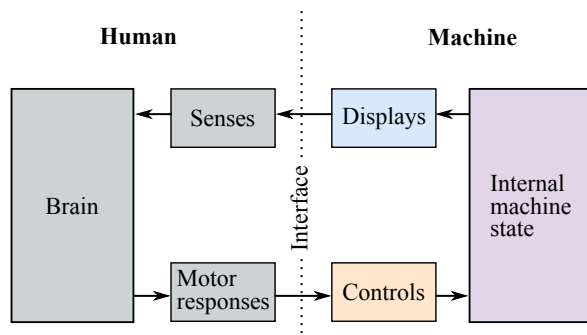
Fig. 1. Human-machine-interface model according to Kantowitz and Sorkin [10]

obstacles) and making it available to the person through an accessible information channel. The person's sensory system (the senses and perceptions available to him or her) takes in both this information and the information perceived directly from the environment (e.g., a car horn) and uses it to construct a *mental environment model* in the brain. The person can then influence the environment through their motor skills and, by interacting with the mobility aid, control its machine state and the *digital environment model* based on it. Other external information channels (e.g., online weather services) can also be used to enrich this digital model.

Looking into ETAs, the following system components, which are directly linked to the internal machine state, can be identified:

1) *Sensors*: Used to gather information from the immediate environment to build up an internal system state.
2) *Controls*: Used to directly control the assistive device without having to go through the environment.
3) *Additional Information Sources*: Sources of information not associated with the system itself, but which contribute to the construction of the internal state of the system.
4) *Displays*: Used to present information to the user, for visually impaired people to substitute the sense of sight, usually in acoustic or haptic form.

An overall system thus represents a composition of concrete instances of these components and a kind of business logic that receives information from *Sensors*, *Controls* and *Additional Information Sources*, converts it into a digital environment model thanks to certain algorithms, and provides a representation of it via *Displays*.

By defining good and consistent interfaces for individual component types, there are two advantages to such a component-based view. On the one hand, you can achieve easy *interchangeability* of individual components without having to adapt much to the overall system. Consider, for example, a navigation system that uses GPS to determine the current position of the user. Developing the same navigation system, but using RFID technology, would now require very little overall effort with a common interface, since only the sensors component would need to be changed. On the other hand,

reusability increases with different overall systems that use the same subcomponents. As an example, consider an obstacle detection system and a navigation system, both of which use a vibration belt as a display component. The former uses it to signal obstacles in a particular direction, and the latter to indicate the direction of travel. If developed within a component-based framework, it would only be necessary to determine the obstacle or walking direction from the digital environment model, but not to redevelop the vibration belt as a component.

When developing mobility aids, avoiding collisions with obstacles, following certain navigation routes or, more generally, minimizing dangerous situations play a crucial role. However, the testing of such dangerous situations is essential for the evaluation of the developed prototypes, which is why a *simulation environment* has great advantages in the development of ETAs. On the one hand, it increases reproducibility by allowing test persons to be led through the same scenarios and their behaviour to be recorded and statistically evaluated. It also increases variability, as a simulation environment can be freely parameterized and configured to meet a wide range of system requirements. For example, weather conditions, which often strongly influence the behaviour of a sensor- or camera-based ETA, can be changed with little effort. It is also possible to generate custom obstacles, roads, traffic situations, etc. Such variability is difficult to achieve in the real world. In addition, the dangerous situations mentioned above can be mitigated, as real collisions are impossible or can be provoked for testing purposes in a controlled environment.

Considering that individual components are to be used in a simulation environment with little effort, it makes sense to embed this environment in the model shown in Fig. 2. In principle, any of the components on the ETA side can be simulated, the most obvious being the physical environment and the sensors. The former is a *virtual reality* in the simulation, which requires it to be sensed by *virtual sensors*. Since such sensors can provide perfect, noise-free environmental data, it is possible to test displays, controls and the algorithm used to build the internal state of the machine individually and in a controlled manner. In the indoor navigation system presented in the next chapters, this is demonstrated in more detail using an example.

Looking at past research projects on ETAs, one can see the presented component-based structure in many of these overall systems, mentioned by Khan et al. in their literature review [2]. Often the boundaries between the individual components become blurred because they are very closely related, but the basic structure remains the same. This suggests that, again, components could be easily exchanged and reused in similar systems if they were developed within a standardized, common ecosystem.

One such component-based ecosystem is ROS 2, which is a set of software libraries and tools for developing applications that originated in robotics (especially mobile robotics). It is open source and aims to support developers from different industries from research to prototyping, deployment and
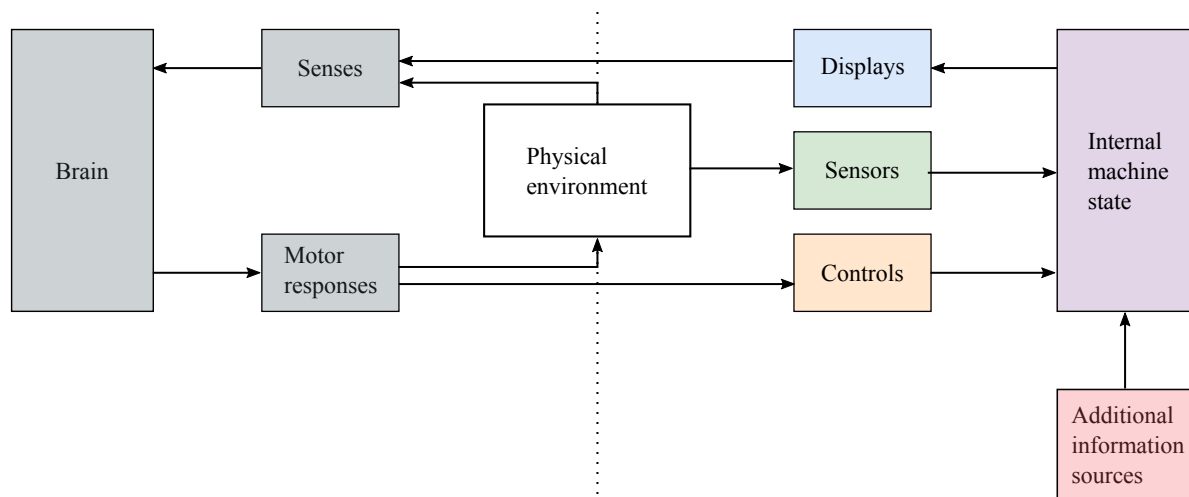
Fig. 2. Extended HMI model for ETAs

production using a standard software platform. The modular and flexible architecture allows easy integration of different hardware and software components, enabling the development of complex overall systems. A standardized real-time capable communication protocol enables efficient and reliable communication between different subcomponents of a system. It is not tied to a specific platform, nor is it domain or vendor specific. Because of its origins in mobile robotics, it provides many algorithms and sensor drivers to address problems of environmental perception, navigation and orientation, problems that are also common in the field of mobility aids. ROS 2 simplifies the development and testing of complex systems by providing debugging, visualization and, above all, simulation tools.

In ROS 2, development is strictly based on the "divide and conquer" principle by providing the following architectural components [6]:

1) *Nodes*: Independent processes that communicate with each other through different mechanisms.
2) *Topics*: Named event channels that allow nodes to communicate with each other. Nodes can publish messages to a topic, and other nodes can subscribe to that topic to receive those messages. Topics can have multiple publishers and subscribers, making it possible to build complex communication patterns between nodes.
3) *Services*: Remote procedure calls that allow nodes to request a specific task or information from another node in a synchronous way. Nodes that provide services and respond to requests are called servers, while nodes that request services are called clients.
4) *Parameters*: Parameters are used to store configuration data for nodes. Parameters can be set and retrieved by nodes, and they can be changed dynamically during runtime.
5) *Launch files*: Used to simplify the process of starting and configuring a ROS 2 system by specifying a collection

of ROS 2 nodes, their parameters and other configuration details without having to start each node individually and configure it manually.
6) *Packages*: Collection of nodes, configuration and launch files and documentation, representing a subcomponent of a ROS 2 system. They provide a modular and extensible way to organize and distribute code, making it easier for developers to share and reuse code across different projects.

This architecture divides a system into a set of intercommunicating nodes, which are in turn organized into packages, providing a modular and extensible way to organize and distribute code, making it easier for developers to share and reuse code across projects. Beneath others, defining standard interfaces and the component-based development made it possible to build up a large and active community that constantly extends ROS's vast array of code libraries, hardware drivers, documentation and support. The community supports a continuous exchange between scientists and developers of new products.

V. PROOF OF CONCEPT

Our proof of concept represents an indoor navigation system developed specifically for blind and visually impaired individuals.

A. Indoor navigation system

The following components were used in this particular use case.

- *bHapticsX40 vibration vest by bhaptics® to provide haptic feedback for navigation instructions.*
- *An ultra-wideband (UWB) real time location system (RTLS) by Pozyx® to determine the indoor position and orientation of a person using anchors placed in the room and a tag attached to the person.*

- *Smartphone App* for configuring the system and for recording and navigating along routes. Its compass feature can also be used as an alternative to the Pozyx tag for providing orientation information.
- A *Raspberry Pi 4* is used for computing operations such as handling services for route recording, providing heading correction for navigation instructions and feedback generation through vibration modes.

To operate the system, all components must be connected to the same network. The Raspberry Pi serves as the primary hub for most of the nodes required in the ROS 2 ecosystem. The vibration vest, with a Pozyx tag attached, can transmit its current position and orientation data to the Raspberry Pi.

A smartphone app provides necessary communication interfaces to the ROS 2 ecosystem, allowing the user to change the parameters of the system and so reconfigure it at runtime. In addition, he can record new routes and navigate along already recorded ones. When a route is selected, the navigation system running on the Raspberry Pi estimates the nearest navigation point available on the route and calculates a heading correction based on the real-time data from the Pozyx tag. This heading correction is translated into an appropriate vibration pattern on the vest to indicate the direction, in which the user should move for safe navigation along the predefined path. As an alternative, audio feedback displayed over the headphones connected to the smartphone can be used for navigation instructions.

The system can be adapted to the model presented in section IV by breaking it down into its components. Here the vest takes on the role of the *display*, the Pozyx tag or the smartphone compass corresponds to the *sensors* and the remaining features of the smartphone application represent the *controls* part. The *internal machine state* is formed by the navigation algorithm and business logic running on the Raspberry Pi. The part of the business logic responsible for recording, persisting and retrieving routes can be seen as an *additional source of information* that enriches the internal machine state and the digital environment model it contains (see Fig. 3).

To demonstrate the practical use of a simulation environment in relation to the development of ETAs in general and specifically with ROS 2, the simulation tool *CARLA*, which is widely used in autonomous driving research, was used. It is also open source and, in addition to existing maps, actors and assets, allows the creation of custom scenarios and the free configuration of environmental factors such as weather and lighting conditions. It also offers a range of different virtual sensors such as LIDAR, cameras, GPS, etc. However, the biggest advantage for the concept proposed in this paper is CARLA's built-in integration with ROS 2 via a bridge. Using predefined ROS 2 topics, it is possible to both read simulated sensor data and control the movement of virtual actors such as pedestrians. In the use case presented here, CARLA replaces the indoor environment and the Pozyx system for determining position and orientation (see Fig. 4). This makes it possible, for example, to test the display components separately without

having to deal with sensor inaccuracies or the influence of a test person's behaviour.

Now we will look at the development process and architecture of this system and how ROS 2 supports it and helps to achieve component-based reusability and interchangeability.

B. Development process

Starting with the core functionality, navigation, the necessary nodes, inputs and outputs were defined. The central node provides a single output, a heading correction value. To provide correct and up-to-date values, it requires the route to be followed and the current position and orientation of the user. Inputs and outputs lead to the definition of their respective interfaces and the nodes that provide the necessary inputs. This means that the navigation logic consists of three nodes and has five interfaces (see Fig. 5).

1) *Navigate route* (action), provided by the navigation and called by the user over the smartphone app
2) *Position* (topic), provided by the Pozyx RTLS or the CARLA simulation
3) *Orientation* (topic), provided by the Pozyx RTLS, the smartphone compass or the the CARLA simulation
4) *Load route* (service), provided by the Route Management and called by the navigation logic
5) *Heading correction* (topic), provided by the navigation system and consumed by the feedback device (bHapticsX40 or headphones)

The result is a fully functional navigation system with a freely configurable setup of sensing and user interface devices – even swapping devices on the fly is possible. Each device requires its own node or set of nodes to transfer data to and from the ROS 2 ecosystem and to satisfy the interfaces owned by the navigation service. For example, the software for the bHapticsX40 vest currently consists of two nodes: A driver node responsible for connecting to the vest via Bluetooth, and a feedback node that translates the heading correction feedback into different motor patterns (see Fig. 6).

To enhance usability beyond direct control via command line terminals, a user interface application must access a service client node. The interaction between the user interface and the service client node is the least clean implementation detail, as ROS 2 does not inherently support direct user interaction.

As the manual creation of routes as sets of coordinates was rather tedious and error-prone, the second service, for route recording, was conceived. It allows the user to record their current location and save it as a route for later retrieval by the navigation service. This route recording service was easily implemented using the existing nodes for the navigation service and proved to be a significant improvement over manually entering coordinates. At this stage, control of both services was limited to launching the required nodes with a set of parameters. To increase control and make it more dynamic, separate control nodes with additional user interfaces were next designed and implemented. As the number of nodes and possible configurations increased, it became necessary to
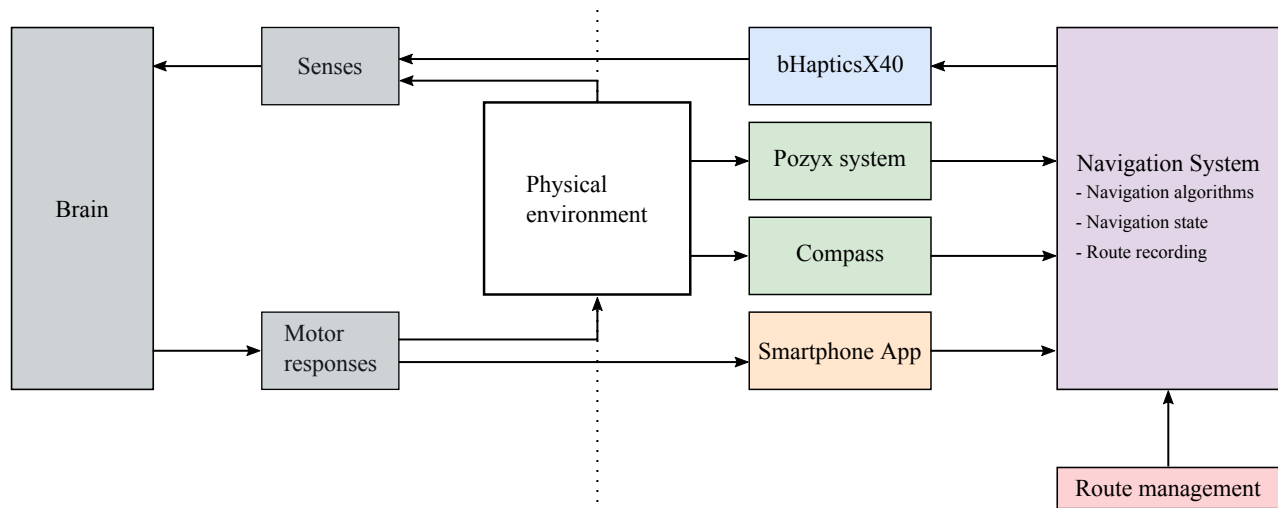
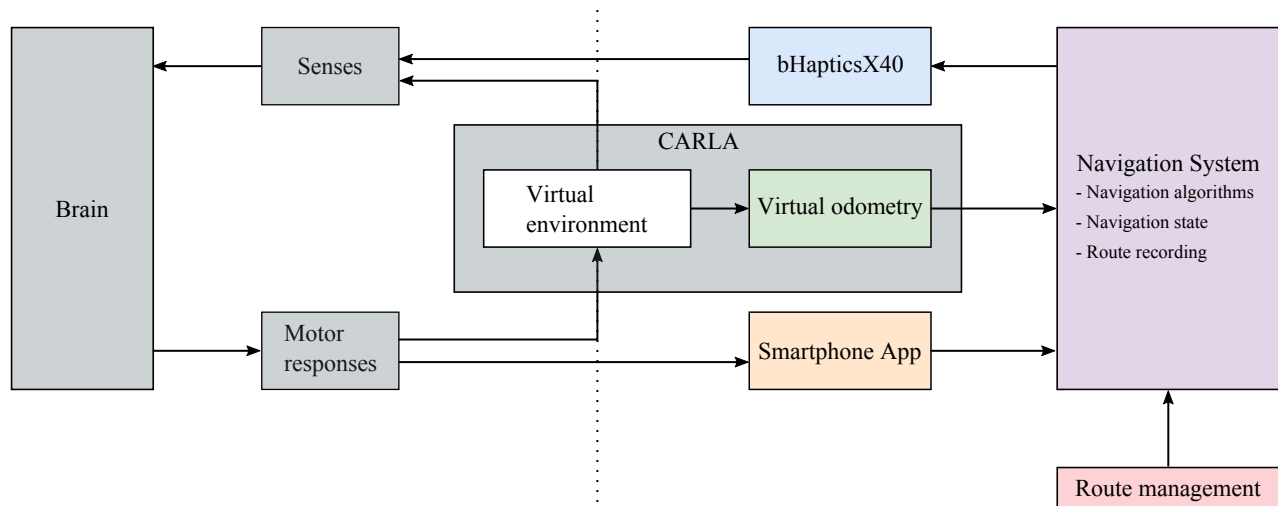Fig. 3. Indoor navigation system according to Fig. 2

Fig. 4. Indoor navigation system with CARLA simulation as environment and sensors substitution

organize the startup configurations using a modular system of ROS 2 launch files. A semi-automated deployment method allowed different distributions of nodes among hardware components to be tested. The actual (graphical) user interfaces in the form of the smartphone app were the last components to be implemented.

C. Architecture

The resulting architecture follows a microservices approach. For example, the existing system with two services, the navigation itself and a utility for recording the route, can be easily extended, both by adding new types of services and by redundancy of existing ones. This guarantees the degree of scalability and elasticity required by possible use cases, such as indoor navigation in public buildings.

The internal structure of the existing services has many similarities. Both consist of controller, business logic and helper components realized by ROS 2 nodes. The controller nodes provide the user-facing interfaces necessary to control the services and translate ROS external user input for the ROS 2 system. The business logic nodes produce the service functionality, possibly with the help of utility nodes. They interface with the controller nodes via ROS 2 interfaces, i.e., actions and services. By structuring services as a collection of nodes, a single service can be distributed across several separate hardware systems if a specific use case requires it. If this flexibility is not required, the nodes of a service can be run on a single system and configured to run in shared-memory mode to optimize performance. The trade-offs can be considered on a case-by-case basis without changing the node implementation.
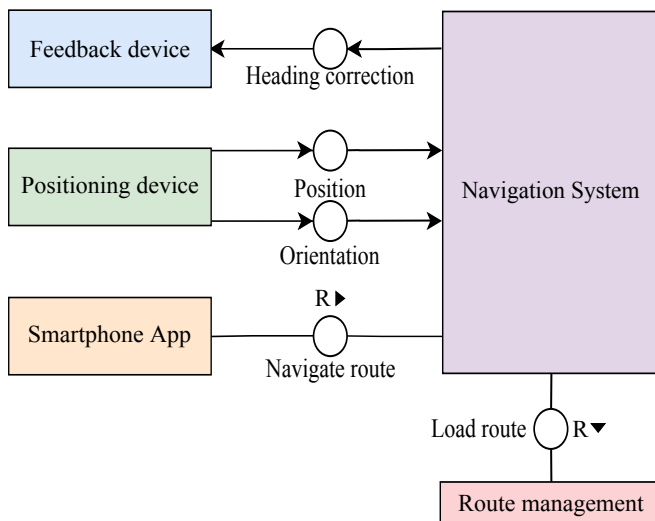
Fig. 5. Abstract FMC model [7] of a navigation system according to Fig. 2

However, ROS 2 supports modularized development of reusable components not only at the level of entire components of the HMI model, but also within the individual components. Fig. 6 illustrates this clearly. It shows the structure of the bHapticsX40 component as a composition of two ROS 2 nodes and the physical vibration vest. The separation into a ROS 2 driver node, which handles the actual Bluetooth communication with the vest, and a feedback generation node, which transforms the received heading correction into a concrete vibration pattern, again promotes loose coupling of the software components. While the former is independent of the concrete overall system and can therefore be reused at any time, the latter can be exchanged in the presented project depending on, which of the two feedback representation modes is used. If, on the other hand, you were to use a similar vest made by a different manufacturer that uses a Wi-Fi connection instead of Bluetooth, replacing the driver node would allow you to quickly reuse the entire system.

The general trade-offs of the chosen architecture can be summarized as follows: Future requirements for new additional functionality, scalability and elasticity can be easily met. Components, especially sensing and user interface devices, can be added and replaced at low cost. Performance is limited by the degree of distribution chosen for a particular deployment. Even with only two services, the actual implementation is structurally and operationally complex.

VI. RESULTS AND EVALUATION

Looking at the architecture of the indoor navigation system just presented, one can see the benefits of reusability and interchangeability of individual components presented in section IV for the HMI-ETA model. The bHapticsX40 vibration vest, which acts as a *displays* component and is represented by a package in ROS 2, will serve as an example. The navigation package provides an azimuth and elevation angle via the heading correction topic, which is consumed by the bHapticsX40 package. By defining this asynchronous data channel independent of the display component in the form of a ROS 2 topic, a low degree of coupling is achieved between the package providing the navigation feedback and the package consuming it. This loose coupling allows interchangeability on both sides of the data channel. On the one hand, the display component can be replaced by a headset package, for example, which communicates the feedback to the user acoustically using a text-to-speech algorithm. In this case, both the topic and its message format, as well as the navigation package, could be reused. On the other hand, the indoor navigation of pre-recorded routes could be replaced by any other navigation package (e.g., outdoor navigation using GPS and an external map service), as long as it respects the heading correction topic as an interface to the display component. Again, neither the data channel nor the display component (whether vest or headphones) needs to be touched.
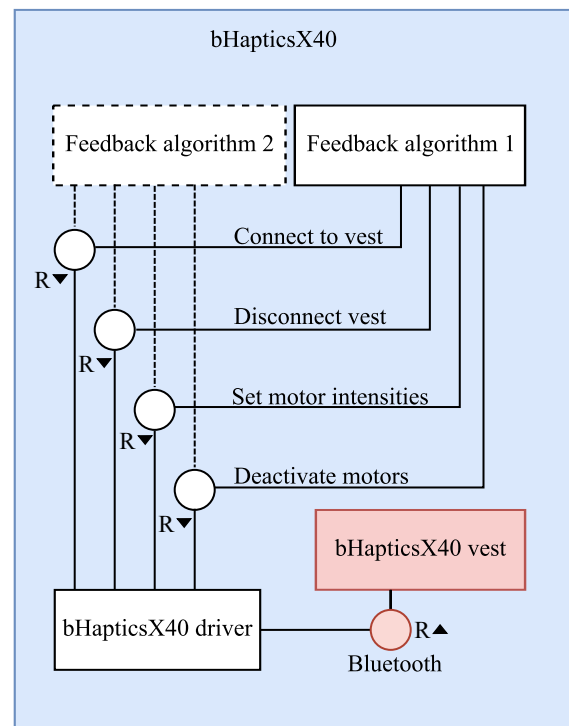

Fig. 6. FMC model of the vibration vest implementation regarding exchangeable algorithms. The red parts are visualized for a full overview and not part of the software system. The bHapticsX40 component is a display device according to Fig. 2

In addition to the simple definition of generic communication interfaces and the resulting loose coupling, the interchangeability of components in ROS 2 is also made possible by the definition of launch files. This is where the entire system is assembled and configured using so-called *launch arguments*. Components in the form of packages and nodes can be added to or removed from the system as needed and can be adjusted by specifying node parameters.

Generally speaking, ROS 2 proved to be very accessible.

The first few tutorials provided enough knowledge to design and implement the whole navigation system with the ROS 2 Python API. Designing a distributed system did not require any additional work thanks to the node structure and the topic, service, and action interfaces. This pre-structuring also assisted with the definition of clean interfaces and division of labour within the team. Deployment, together with launch configurations, was less accessible and considerably less well documented. Only the myriad of existing ROS 2 projects and the associated launch files provided any orientation in this regard.

The lack of documentation in some cases may be due to the fact that ROS 2 is an open-source framework. Although the ROS 2 community is usually very active and helpful, there is no guarantee of support compared to proprietary systems, which can make the lack of documentation all the more problematic. However, the open-source status does have some advantages, including complete transparency in the provision of the source code. This means that issues or vulnerabilities can be discovered and addressed more quickly by the community. It also allows everyone to contribute to the development and to share knowledge and expertise, features that could drive forward ETA research. However, it must be recognized that building up a community can be a long and arduous process. Even ROS, which was developed at Stanford University in 2007 and evolved into its now well-known successor ROS 2 in 2015, did not immediately have the reputation it has today and took years to build such a large community.

The types of problems encountered in robotics have many similarities with those encountered in the indoor navigation project. This means that many of the robotics-oriented packages created for ROS 2 could be adapted accordingly. An example of this is the *tf2* package provided by the ROS 2 community, which makes it possible to track the temporal evolution of several interdependent coordinate systems and to perform transformations between these frames in a simple and efficient way. This is an essential component in robotics, as such calculations are the basis for calculating the individual joints of a robot arm, for example. In the indoor navigation project presented here, tf2 was able to help transform coordinates from the global coordinate system of the Pozyx system to that of the person being navigated, and thus determine a heading correction.

For the same reason, there are already some packages for hardware components for ROS 2 that allow the integration of different sensors from different manufacturers, although this is not visible in the proof of concept. Examples include camera, LIDAR or ultrasound drivers that can be used in robotics as well as ETA development without much expertise or training.

Simulation in ROS 2 is also well-supported. Nodes are configurable for a simulated environment without the need for any code changes. The debugging tools within the ROS 2 ecosystem proved to be extremely helpful and easy to use, as well. Not surprisingly, GUI functionality is an aspect not supported within the ROS 2 ecosystem, but various types of bridge tools provide the possibility to access ROS 2 interfaces.

In order to be able to evaluate the assumptions for the evaluation of a system in reality, an evaluation of 4 different user interfaces was carried out. The user interfaces were used in simulation and in reality. This experimental setup is shown in Fig. 7 and Fig. 8. In addition to the existing nodes, there was a metric node during the test execution, which recorded the positions of the subjects every second and determined the deviations from the ideal route. In addition, the times between waypoints of the route and the total time were recorded. In simulation a single route was run, in reality a total of four. Only sighted people were used to perform the test, and they ran the routes in reality with their eyes open so that there was no risk of injury. Future evaluation runs will be made with people who have different visual impairments. In the simulation, the other extreme was chosen and there was no feedback on position or movement, except for statements about whether someone was standing or moving, or rotating around a point. This minimized the perception of one's presence in the virtual environment. According to Witmer and Signer, this has a weak but consistent effect on task performance, the task in this case being the traversal of a route [11]. The four display components are:

Device 1 A bHapticsX40 vibration vest that continuously vibrates in the target direction.

Device 2 A bHapticsX40 vibration vest that vibrates continuously. It vibrates to the left or right depending on the direction, in which a user is to turn, or straight ahead if a user is aligned with the target point.

Device 3 A text-to-speech interface that announces the target direction as the time. Only 12 or 6 o'clock are replaced by the statements "Perfect" or "Turn Around".

Device 4 A text-to-speech interface that aligns a person to the target direction using "Left", "Right", and "Perfect" instructions.

This paper focuses more on the development and evaluation of ETA components than on the results of evaluations of specific ETA components. Therefore, the focus is on the implementation of the evaluation and the lessons learned. During the execution, each test subject had to complete eight scenarios, each resulting from the use of the four different display components in both simulated and real-world environments. Four different routes were used in reality to prevent the participants from knowing them after the first scenario. Therefore, the individual points of a route had to be navigated, which were spread out in space. The simulation utilized only one route due to the lack of feedback regarding the environment or the subject's movements. To implement a scenario, an entire system must be started along with the simulation if required. The availability of ROS 2 launch files can be helpful while starting the whole system. Launch files are utilized to initiate the entire system with all software nodes and the scenario's parameters. This method enhances reproducibility by describing a device's configuration and parameterization in its constellation. It minimizes the susceptibility to errors
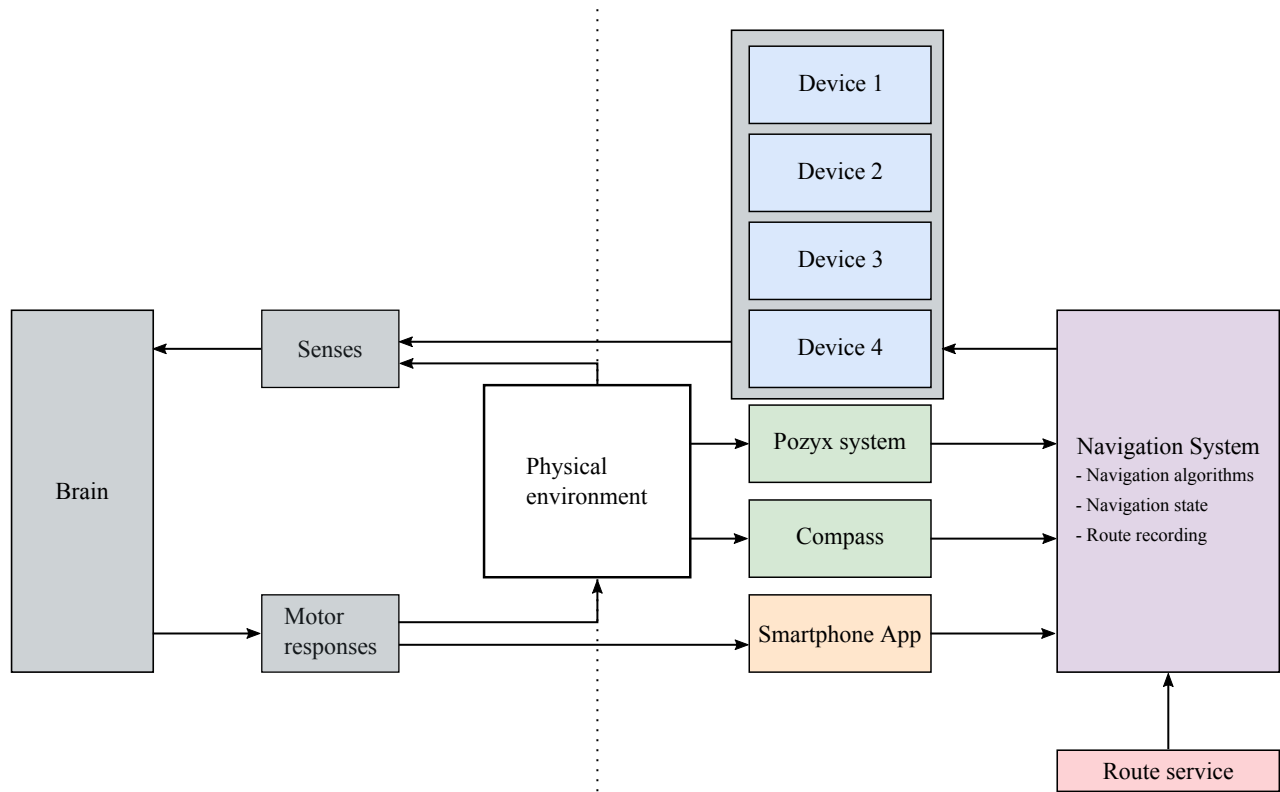
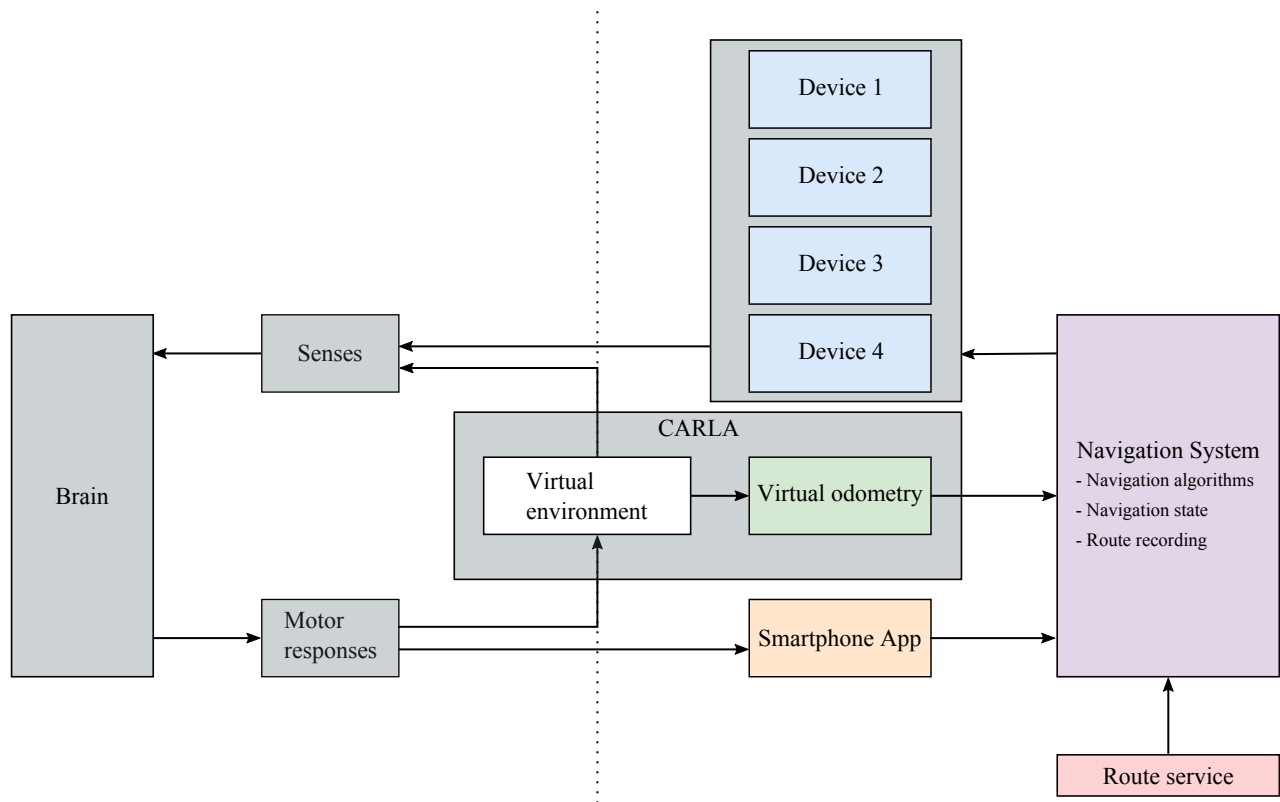Fig. 7. Four alternative user interfaces for a real-world navigation task.

Fig. 8. Four alternative user interfaces for a navigation task in simulation.

that may occur during manual start and parameterization of the components. The participants benefit from reduced waiting times for the preparatory work during a scenario change.

In the experiments, a software node was launched for recording metrics. The software node measures the distance, the user's pose, and the current time. Based on this information, the software node makes a recording every second and writes it to a file, which can be reviewed later. Consequently, information about the user's pose and the distance deviation from the target is available every second. Initially, this data is used to compare devices or algorithms with each other. Subsequently, the information serves as the basis for observations that can be used to enhance the execution of experiments or achieve better results.

Further findings suggest that statements regarding the use of assistive devices in simulation are initially incomparable to their real-world use. This may be attributed to the experimental setup, i.e., participants could work in the real world with full sensory support but relied entirely on the assistive device's feedback without any other sensory perception in simulation. Feedback on the degree of turning can be highly beneficial, particularly when comparing the effectiveness of device 2 compared to device 1 in terms of distances and times per meter, to reach the waypoints of the route (see Fig. 9). The current angle of orientation is represented in device 2 as $\pm 22.5°$. Since the participants could not perceive the angle of rotation in the simulation, the midpoint between these $\pm 22.5°$ was often not determined correctly. Incorporating the results of Device 3 and Device 4 could expose a contradiction at this point. Nevertheless, based on a survey of the participants, it became evident that processing the time information required additional effort, contributing to the further complication of this issue. The times between waypoints support these results.

One notable observation made during the implementation process and supported by the data evaluation is the significant difference in freedom of movement between reality and simulation. This can be seen by comparing Fig. 9 and Fig. 10. Examining the distances to the target route, the simulation range of 1-4 meters (see Fig. 9) exhibits less precision compared to the 0-0.9 meter range in reality (see Fig. 10). In reality, a subject has complete control over their speed and orientation. This is illustrated in Fig. 3 through direct interaction and perception with the surroundings. This type of interaction is more restricted, but only if the environment is simulated. At that point, an interface like a keyboard determines, which movement takes place in the simulation. Yet, the simulation, not the subject, has control over the motion as it interprets a keystroke. The statement is equivalent to the argumentation above in terms of necessary feedback. Nonetheless, the effect differs. Navigation and other algorithms rely on data from the virtual environment, where only rough motor movements are possible. Therefore, tolerances must be larger than those in reality. This was particularly noticeable in our case in terms of navigation, as it heavily relies on user behavior. The system only switches to the next point on the path when a point within
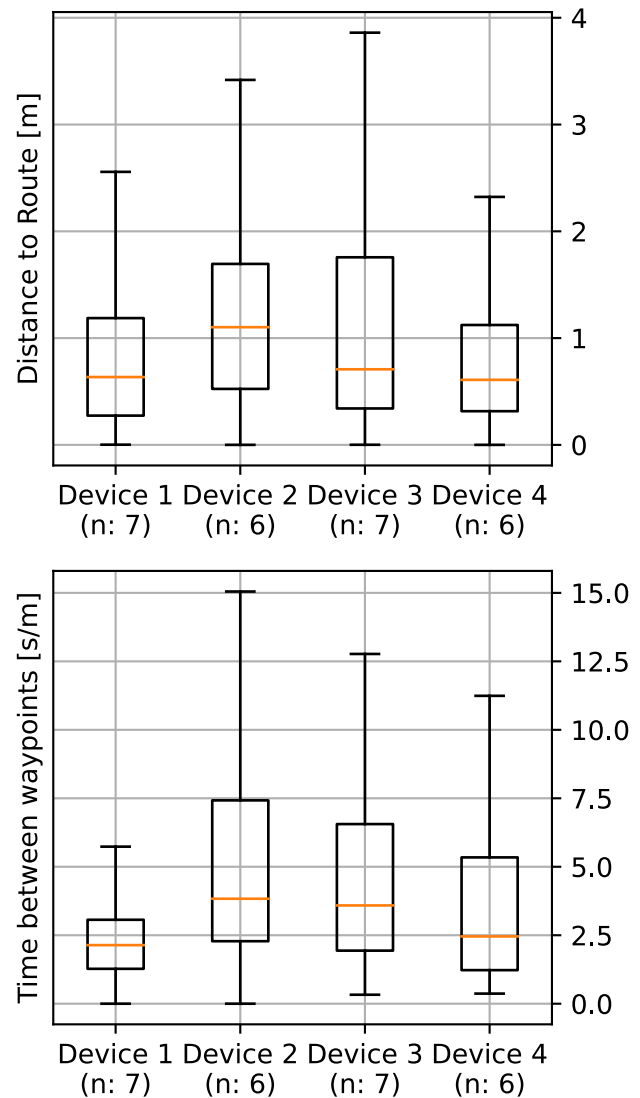

Fig. 9. Simulation results related to the devices. The distance to the route denotes the distance measured towards the designated route every second, expressed in meters. The value of n denotes the number of datasets utilized for the plot. The time between waypoints describes the time spent between waypoints divided by the length between the starting point and the end point.

a radius is reached. While not ideal, this serves as a good example of algorithms that rely on the user's motor skills.

The benefits of interchangeable subsystems in evaluation have been demonstrated. When evaluating display components, a greater level of comparability can be achieved by ensuring identical test setups and data generation procedures. This is necessary because differences in results, despite equivalent interfaces, may be caused by various influences at each level of data generation or even in the test setup. Fig. 7 and Fig. 8 demonstrate that the presented development approach enables a purely interchangeable display component, and that the results are comparable, especially with an unchanged rest of the system. When comparing reality with simulation,
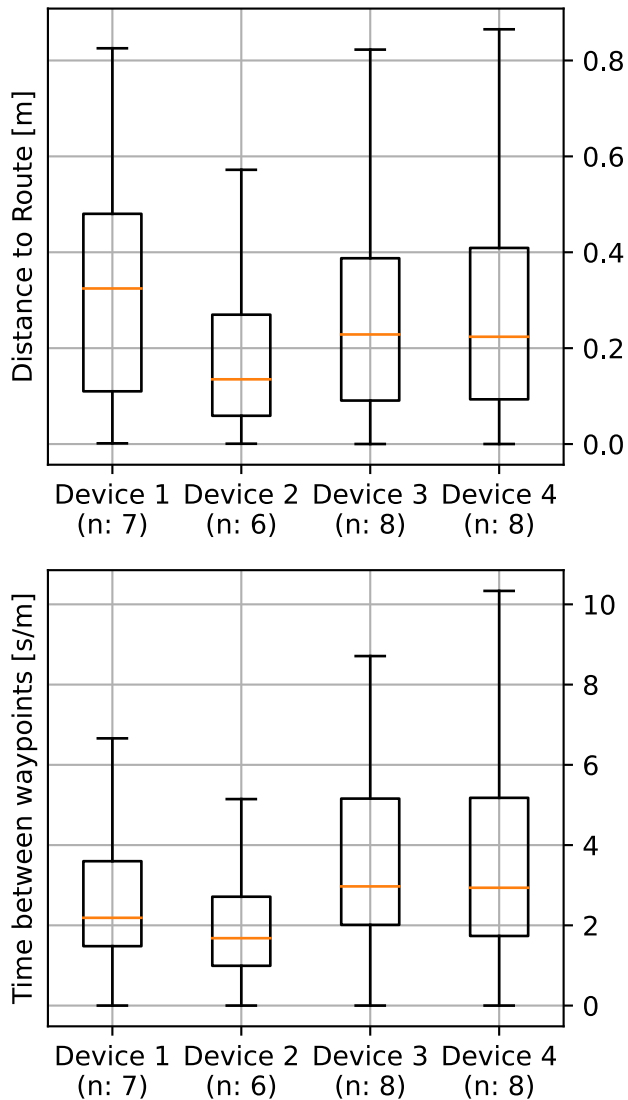
Fig. 10. Real-world results related to the devices. The distance to the route denotes the distance measured towards the designated route every second, expressed in meters. The value of n denotes the number of datasets utilized for the plot. The time between waypoints describes the time spent between waypoints divided by the length between the starting point and the end point.

problems arise due to the strong influence of virtual presence and motor activity on the results.

In summary, ROS 2 provides the necessary building blocks for loose coupling thanks to the provided architecture components such as topics, nodes, packages and launch files, thus supporting a modularized, component-based development of ETAs from the ground up. The challenges and work areas known from (mobile) robotics, which also need to be addressed in ETA development, such as navigation or environmental perception, are facilitated by the tools and drivers already available in ROS 2, allowing most of the time to be spent on the actual development of innovative ideas.

VII. CONCLUSION

While research into ETAs and mobility aids for the blind and visually impaired in general has produced numerous research papers and demonstrators over the past decades, reviews of these technologies show that the wheel is often reinvented. Both the individual hardware components and the algorithms used to generate feedback, among other things, are redeveloped instead of shared. A lack of exchange between research groups and the use of different development ecosystems means that these software and hardware components often have to be developed and integrated from scratch. This takes up valuable resources that are then not available elsewhere, e.g., for developing innovative concepts or testing them with visually impaired users.

This paper presented an approach to this problem by introducing a framework for a component-based development of ETAs that promotes the reusability and interchangeability of components across projects within a standardized ecosystem. To this end, a mode for identifying ETA component borders by using a human-machine interaction view was presented (see Fig. 2). The components were identified as displays, sensors, controls, a machine internal state and additional information services. It can be concluded that an ETA is generally suitable for decomposition into loosely coupled building blocks. This subdivision can also be seen in systems already known from related literature. Furthermore, individual components can be replaced by a simulation, allowing certain other components to be tested in a more flexible and risk-free manner.

ROS 2 was proposed as an existing open-source framework to support the component-based development of ETAs. The development with ROS 2 and the component-based development was shown in the form of an indoor navigation system. This example uses UWB technology for localization with a vibration vest taken from the virtual reality gaming domain, to present the advantages and disadvantages of such an ETA development. This paper also shows an evaluation of that system and highlights the possibilities of exchangeable feedback devices as well as CARLA as an open-source simulation environment used in autonomous driving. It is highlighted that the results of a simulation highly depend on the feedback available to the user inside the simulation, like when changing orientation or position.

Two features of ROS 2 proved to be particularly important advantages. One is the background of ROS 2, which is mainly in mobile robotics. The overlap between the problems addressed in robotics and ETAs, and the technologies used to address them, is remarkably large. Examples include real-time navigation and environmental perception, for which ROS 2 already provides appropriate sensor drivers, standardized interfaces and algorithms, and tools for testing and visualizing the systems. The use of simulations is also commonplace in robotics.

On the other hand, ROS 2 is designed to support a component-based development. A loose coupling between ROS 2 nodes and packages is enabled by the definition of

custom asynchronous data channels and launch files, in which components can be configured and integrated. This creates the reusability and interchangeability identified in this paper as important drivers of ETA development.

Despite the fact that ROS 2, as an open-source framework, has limitations, such as a lack of documentation or support, it enables rapid results and innovation, not least because of the large and active community, especially for research.

Considering the possible implications for the future development of ETAs using ROS 2, the framework we have started and presented would need to be extended with even more components and system compositions to further illustrate how generic and versatile it is. To facilitate exchange between different research groups, an open platform could be created where components, algorithms and complete systems could be published in the form of ROS 2 nodes, packages and launch files. However, to ensure compatibility between components from different developers and systems, a more concrete policy for their creation needs to be formulated by defining rules and specifications. Then new components could be easily integrated. Although, experience with other open-source frameworks has shown that building a community to collaboratively share knowledge and expertise can be challenging.

REFERENCES

[1] F. v. Zabiensky, G. Fridman, S. Reuter, O. Özdemir, M. Kreutzer, and D. Bienhaus, "A Framework for Developing Modular Mobility Aids for People with Visual Impairment: An Indoor Navigation Use Case," presented at the SMART ACCESSIBILITY 2023, The Eighth International Conference on Universal Accessibility in the Internet of Things and Smart Environments, Apr. 2023, pp. 10–18.

[2] S. Khan, S. Nazir, and H. U. Khan, "Analysis of Navigation Assistants for Blind and Visually Impaired People: A Systematic Review," IEEE Access, vol. 9, pp. 26712–26734, 2021, doi: https://doi.org/10.1109/ACCESS.2021.3052415.

[3] M. D. Messaoudi, B.-A. J. Menelas, and H. Mcheick, "Review of Navigation Assistive Tools and Technologies for the Visually Impaired," Sensors, vol. 22, no. 20, p. 7888, Oct. 2022, doi: https://doi.org/10.3390/s22207888.

[4] B. Kuriakose, R. Shrestha, and F. E. Sandnes, "Tools and Technologies for Blind and Visually Impaired Navigation Support: A Review," IETE Technical Review, vol. 39, no. 1, pp. 3–18, Jan. 2022, doi: https://doi.org/10.1080/02564602.2020.1819893.

[5] M. Hersh, "Wearable Travel Aids for Blind and Partially Sighted People: A Review with a Focus on Design Issues," Sensors, vol. 22, no. 14, p. 5454, Jul. 2022, doi: https://doi.org/10.3390/s22145454.

[6] 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.

[7] S. Wendt, "FMC: Home of Fundamental Modeling Concepts," http://www.fmc-modeling.org/ (accessed Dec. 12, 2023).

[8] M. Quigley et al., "ROS: an open-source Robot Operating System," in ICRA workshop on open source software, 2009, vol. 3, no. 3.2, p. 5.

[9] K. Wyrobek, "The Origin Story of ROS, the Linux of Robotics," IEEE Spectrum, Oct. 31, 2017. https://spectrum.ieee.org/the-origin-story-of-ros-the-linux-of-robotics (accessed Dec. 12, 2023).

[10] B. H. Kantowitz and R. D. Sorkin, "Human Factors: Understanding People-System Relationships," 1st edition. New York Chichester Toronto: Wiley, 1991. ISBN: 978-0471095941.

[11] Bob G. Witmer, Michael J. Singer, "Measuring Presence in Virtual Environments: A Presence Questionnaire," Presence: Teleoperators and Virtual Environments, vol. 7, no. 3, pp. 225-240, June 1998. doi: https://doi.org/10.1162/105474698565686.