UNIVERSITY OF TWENTE.

Development of an Eye-controllable Self-Driving Wheelchair

A Master's thesis by Sjoerd de Jong

Interaction Technology, EEMCS

19-07-2023

Supervisors:
dr.ir. Edwin Dertien
dr. ir. Douwe Dresscher
dr. Khiet Thruong

University of Twente

Acknowledgements

I would like to thank my supervisor Edwin Dertien for his continuous support throughout the project.
I would also like to thank Andrei and his father Arie Fokkink for working with me on this project.

Abstract

This graduation thesis involves the design and implementation of an eye-controlled, self-driving wheelchair for people who are unable to control conventional, often joystick operated, wheelchairs. The wheelchair is developed in co-creation with a fully immobile person and his father. Literature reviews show existing research on eye-controlled wheelchairs, but current implementations require tedious user inputs. There are many examples of indoor mobile navigation systems, however, their applicability to wheelchairs is limited. Therefore a need for a self-driving wheelchair with minimal required eye input, including adhering to strict safety requirements, was established. Suitable hardware was chosen and implemented, as well as software in ROS for controlling the wheelchair. Furthermore, different eye-tracking based GUIs were designed and tested to determine the best way to use eye tracking for wheelchair control. All elements were combined into a final prototype, of which its validity was proved with user tests. While the user was able to use the prototype to move himself around, full self-driving capabilities were not achieved, and thus more research is needed to improve the self-driving abilities of the wheelchair.

Contents

1. Introduction

1.1 Motivation and scope

People with disabilities often experience difficulties in their daily lives. Activities of Daily Living (ADL) like washing, dressing, eating and using the toilet, which are routine tasks that most young, healthy individuals can perform without assistance, are for others a daily battle. Furthermore, people with (physical) disabilities in some cases lose one of their most important social rights; the right to move. The physically disabled are often wheelchair-bound and unable to perform the daily activities that others deem normal, and in many cases can make them feel socially excluded.

In more severe disabilities, like locked-in syndrome (LiS) [1], patients are not just wheelchair bound, but also completely dependent on caregivers. An individual with LiS is in most cases not cognitively impaired, but does not have control over any part of their body except for their eyes. The inability to communicate and perform ADL's can reduce the individuals quality of life, and also make it very difficult for them to actively be a part of the community.

Assistive technologies can give disabled individuals more freedom and increase their abilities to perform certain tasks. They can reduce the impact the disability has on their lives and improve their wellbeing. Assistive technologies can be any item, piece of equipment, software program or product system, and can be anything from simple things like cardboard communication boards, to prosthetics, head trackers and computer programs.

The main challenge with developing assistive technologies, however, is the wide variety of disabilities and problems for which they are needed. Each person may have different abilities and disabilities, along with different ADL's they struggle with. Building a gaming interface for an individual with duchenne may be completely different from building a gaming interface for an individual with LiS. This poses a challenging task for developers of assistive technologies, as each product they design often has to be tailored to only a small minority of users, and more often than not, tailored to single individuals.

This research will focus on researching, developing and testing a self-driving wheelchair, as well as an user interface that can be controlled using eye-tracking. The wheelchair will be developed in co-creation with a patient with locked-in syndrome and his father, who set up a platform called Ability Tech (https://abilitytech.nl/). Ability Tech aims to develop innovative, mostly technology-based solutions for problems that less abled people are struggling with. The wheelchair will be developed for Andrei Fokkink, a 26 year old man (also see https://www.andreitekent.nl/). While Andrei is not able to control any part of his body, he is able to fully control his eyes, which he also uses to control a gaze-tracking tablet which he uses to communicate and for entertainment.

The main goal of this project is to develop a (semi-)self-driving wheelchair that allows him to drive to a desired location using his eyes.

1.2 Research questions

The research questions that will be addresses in this project are:
Main RQ:

- *“How can we develop and evaluate an eye-controllable, self-driving wheelchair for (partly) immobile people?”*

Sub RQ:

- *“Which method of input controls suits both the user and application the best when normally functioning muscle control is absent? (eye-tracking, brainwaves, etc.)”*
- *“What are the best methods of making the wheelchair environment-aware and what software algorithms are needed? (Lidar, 3D camera, etc.)”*
- *“How can an interface be developed that will both be efficiently usable by the user, as well as provide enough input for the driving algorithms?”*

1.3 Approach and outline

This report will start by performing a literature review to get a better understanding of the problem statement, related work, ethical and liability implications, and stakeholders. After that, interviews will be carried out to obtain more specific requirements from the stakeholders. Thereafter, the wheelchair hard- and software will be designed, implemented and tested. The report will conclude by stating a conclusion and discussion, and proposing further work.

2. Background / Related work

2.1 Locked-In Syndrome

Locked-in syndrome (LiS) is a rare condition caused by damage to the brainstem, most commonly by a stroke, hemorrhage or trauma. Patients affected by LiS retain consciousness, but are unable to produce any or very little voluntary movements (apart from eye movement), as well as the inability to speak. It can occur in both males and females at equal rates and at all ages. Affected individuals have to rely completely on caregivers. [1]

While cognitive function is usually unaffected, their attention, executive function, intellectual ability, perception, and visual and verbal memory can in some cases be affected as well. Patients are still able to hear and see, and in some cases the patient is able to feel physical pain, but this depends on the form of LiS. [2]

Someone with LiS is unable to control their voluntary muscles, apart from the muscles controlling the eyes. Some individuals can only move their eyes up-and-down (vertically), but not side-to-side (horizontally). In most cases, individuals can blink as well. [3] In a majority of cases, the patient can still produce (unarticulated) sounds, but they cannot speak or control these sounds in any meaningful way. [2]

There are 3 main forms of locked-in syndrome: [1]

- *Classic*, where the individual has complete loss of motor functions but has preserved consciousness and vertical eye movement.
- *Incomplete*, which is the same as classic but with remnants of voluntary movement apart from vertical eye movement.
- *Total*, which involves total immobility and inability to communicate. (no eye movement either)

Each form of LiS has its own complications, quality of life and requirements. An individual suffering *total* LiS is not able to communicate with the outside world at all, except for possibly brain waves. This leads to many misdiagnoses from LiS patients as being in a vegetative state or other, and can have a negative effect on the recovery of the patient. Diagnosing patients with *Classic* or *Incomplete* forms is less prone to errors, but due to the rarity of the condition, can still go misdiagnosed for a while as well if no one notices the eye movements. [3]

2.2 Existing assistive technologies

2.2.1 Communication

There are various existing assistive technologies that help fully or partially immobile patients to communicate with the world. The technologies used depend on the abilities and disabilities of each individual patient. In LiS, communication is usually done using eye-movement. If the patient has more control over their body, things like vocal sounds or muscle movement can be used. In extreme cases where even the eyes cannot be used, patients may still be able to communicate using brainwaves.

The most simple way of communication using eye-movement is to, for example, look up or down, or to blink once or twice, for ‘yes’ and ‘no’. This is a good way of being able to ask simple yes/no questions and receive a quick answer back, but does not allow the patient to fully express themselves.

One of the most low-level assistive technologies for communication is the Letter Board [4], shown in figure 1, with which patients can construct words and sentences by using their eyes or other signals to confirm whether to select a certain row/column pointed at by a caregiver. The caregiver calls every color, “red”, “yellow” … until the patient blinks, looks up, makes a sound or other. This is then repeated for every letter in that row. Although this method is slow and intensive for the patient, this allows them to communicate and express themselves. Digital versions of the Letter Board exist already, allowing patients to communicate without the need of the presence of a caregiver, and also speeds up the process by utilizing auto-complete. [5]

A	B	C	D	End of word	
E	F	G	H	End of sentence	
I	J	K	L	M	N
O	P	Q	R	S	T
U	V	W	X	Y	Z

Figure 1: Letter board

In recent years, more high-end solutions have been developed. For patients who have more or less full control over their eye-movement, special tablets with eye-trackers can be used. Tobii Dynavox products [6] are widely used, and are also used by the person for which the solution will be made in this project. Eye-tracking tablets allow people to do almost anything that can be done with a regular mouse, from typing on a virtual keyboard, to surfing the web, making music and drawing. Because of small inaccuracies in the eye-tracking hard- and software, and the inability

for patients to keep their eyes a hundred percent focussed and steady for a prolonged period of time, however, the tablets usually come with their own user-interface with bigger buttons, and extra confirmation steps to increase the accuracy of the user input. An example of such an eye-tracking tablet is shown in figure 2.

Figure 2: Tobii Dynavox eye-tracking tablet

2.2.2 Wheelchairs

People with physical disabilities often have trouble moving themselves around. A wheelchair can be utilized to increase the mobility of physically challenged people. Normal, non motorized wheelchairs are most commonly used by people who are either able to push forward themselves, in cases where they retained enough control and power over their upper limbs, or by caregivers to move lesser-abled people to their destination. [7]

However, operating a regular wheelchair can be quite difficult for a patient, even when they still have control over their upper limbs, since it requires a lot of muscle strength and endurance. Wheelchair users also have an increased risk of upper-limb extremity injuries compared to non-wheelchair users, due to repetitive loading of the upper limb. [8] In cases where the person is not able to propel himself forward, they have to rely on a caregiver. This can both lead to social exclusion for the patient, and increased stress/pressure for the caregiver. [7]

To fix this, an electric wheelchair can be used. Usually, electric wheelchairs consist of two independently moving wheels with two non-driven wheels, along with a wireless power source and a joystick for control. Electric wheelchairs greatly reduce the loading of the upper-limbs and increase the independence of the user, giving them more freedom on where and when they want to move around. [8]

Wheelchairs can come in many different shapes and sizes, in some cases tuned specifically for the end user. Depending on the specific needs and requirements

of the end user and the environments and scenarios in which the wheelchair is used, different elements of the wheelchair can be adjusted. These include things like the weight of the wheelchair, whether it is foldable or not, or whether or not it has an adjustable chair. [9]

There can also be a variety of different wheel configurations, depending on the specific requirements, as shown in figure 3. Most commonly, electric wheelchairs have two independently moving wheels, along with one or more non-driven wheels for stability. Some smart wheelchairs have also been developed which can deal with more challenging terrains like rocks and stairs, or can maneuver in more narrow spaces by using for example omni-wheels for a smaller turning radius. [9]

Figure 3: Different types of wheelchairs

Some people, like people with LiS, are not able to control the joystick due to limited control of their hand movement, which is required to hold and move the joystick. Users need to keep putting a certain amount of effort to operate the wheelchair continuously until it reaches the destination by using the joystick, which may be challenging even for people who still have some degree of hand control. [8]

Self-driving wheelchairs can in this case be a solution. Self-driving wheelchairs use environmental sensors to map their surroundings, and use that information to drive the user to their desired destination. They require little to no physical input from the user.

2.3 Input methods

An important aspect in self-driving wheelchairs is the ability for users to decide where to drive to, with minimal to no physical effort. Depending on the specific abilities and disabilities of the user, different input methods can be used. Parikh et al. described how the best method for each individual user can be established. [10]

2.3.1 Touch

For users with some degree of control over their upper limbs, a touch screen can be used where the user can either tap or draw a path to the desired destination. This reduces the strain on their hand which would be the case with prolonged joystick driving, to a short, minimal input.

2.3.2 Brain waves

In case a user is not able to sufficiently move their upper limbs, a brain wave interface (BCI) can be used. These devices are placed on the skin close to the brain, and are able to measure and interpret brain activity. BCI technology has come quite far over the past few years. They do however still have their drawbacks with regards to comfortability of the hardware and their accuracy. The model used to interpret the signals also has to be trained on each individual separately, which may not be practical. Abiyev et al. integrated a BCI for the control of a wheelchair using EEG signals. A neural network is used to predict the intended control commands from the user. They stated that they were able to achieve a 100% accurate classification rate using this method. [11]

2.3.3 Cloud

In case the user does not have sufficient cognitive and/or physical abilities to be able to safely control the wheelchair, a remote operator can control and give directions to the wheelchair from a distance. While this cloud-based control works, it still requires a remote operator to be available, and still requires a way for the patient to make known where he wants to move to. [10]

2.3.4 Voice

If the user is able to speak, voice commands like “go left”, “stop” or “drive to the front door” can be used. Even when the patient is not able to produce articulated words, the produced sounds can be analyzed in terms of duration, pitch and other features and used to control the wheelchair.

2.3.5 Other input methods

As Plotkin et al. [14] showed, it is also possible to control a wheelchair using sniffing. Their setup consisted of a sniff sensor, which can detect when a user sniffs in and out, while being able to filter out regular breathing patterns. This was then hooked up to the wheelchair controls, where two sniffs in meant forward, two sniffs out meant backwards, and one-out/one-in and one-in/one-out meant turn left and right, respectively. They proved that it was possible to control a wheelchair using this setup with reasonable accuracy.

Harish et al. [15] developed a wheelchair that is controllable with movement of the tongue, for persons disabled with quadriplegia(paralysis below the neck). An infrared sensor is used to determine the orientation of the tongue, which is used to directly control the wheelchair. The tongue can be moved to the left of the mouth to turn left, right to turn right, and moved forward and backward to drive forward and backwards.

2.3.6 Eye movement

Aforementioned input methods may not be suitable for people who have no voluntary muscle control. Mainly for locked in patients, controlling the wheelchair with eye-movements could be a better solution. Eye-movements can be tracked with a readily available eye-tracking camera. This input can then be used to either control a virtual joystick, clicking buttons for step-based movement, or by looking at the desired destination on a top-view map.

There is existing research towards eye-controllable wheelchairs, like the Eyedrivomatic [12], which is a low-cost solution that uses small servo motors to actuate the joystick of a wheelchair, and uses simple on-screen buttons to indicate the direction. While this works, the wheelchair itself is not aware of its surroundings and thus cannot intervene when the user gives an unintended input, which in turn can lead to unsafe situations. The setup also requires the user to constantly interact with the ui while driving. It uses a step-based control, where the user can click buttons to make the wheelchair e.g. drive forward for 5 seconds or turn left for 2 seconds. Especially for longer distances, this is not ideal and can become discomforting, as after every few meters, the user has to interact with the screen again. The hard- and software of the Eyedrivomatic can be seen in figure 4.

An alternative solution is the one proposed by Dahmani et al. [13], which does not make use of any visual user interface, but rather takes the point where someone is looking at and drives to there. Using this gaze data, they showed that it was possible to control a wheelchair this way. The main challenge with this solution, however, is to distinguish between normal eye-movement and intended driving inputs of the user. E.g. not everything you look at is a waypoint. When you drive around, you may want to

be able to look around, for example to scan for obstacles or have a conversation, and this should not cause the wheelchair to move into that direction.

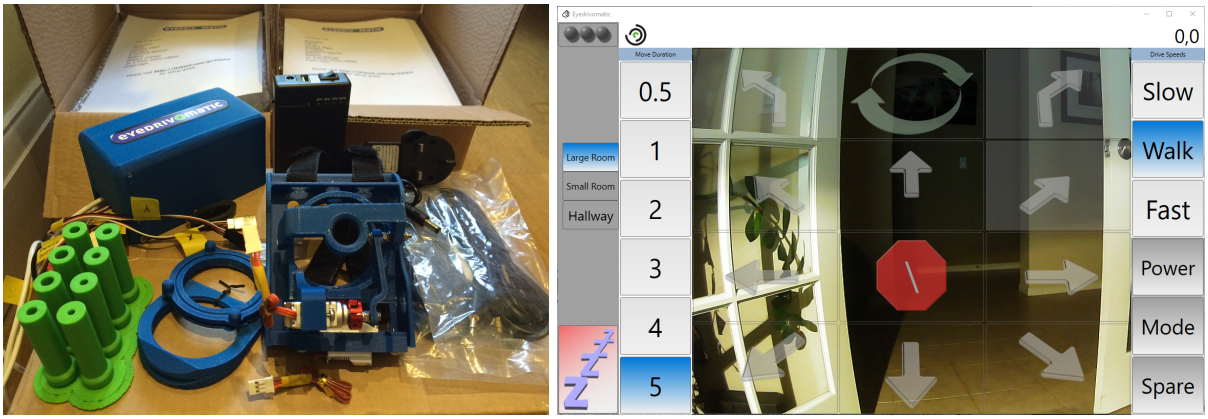

Figure 4: Eyedrivomatic hardware kit and software interface

In order to know what the user is looking at, a suitable eye tracker has to be used. An eye tracker is able to deduce where the user is looking at in 3d space, just from the observed position of the pupils in the eyes. Most modern eye trackers use near-infrared beams along with a high resolution camera. First, a simple face tracking algorithm is used to estimate the position of the eyes, after which the infrared beams are directed to shine onto the pupil. A camera is then capturing these reflections, and using calibrated user data and complex algorithms, it is able to extract the gaze data. All of this can be done in a non-intrusive way from a nearby placed module, without the need to attach anything to the user. [20] This method is visualized in figure 5.

There are many commercially available eye-tracking systems which can do this all for us. For this project, a Tobii eye-tracker will be used of which its input can be integrated like a normal mouse, although with less precise movements.

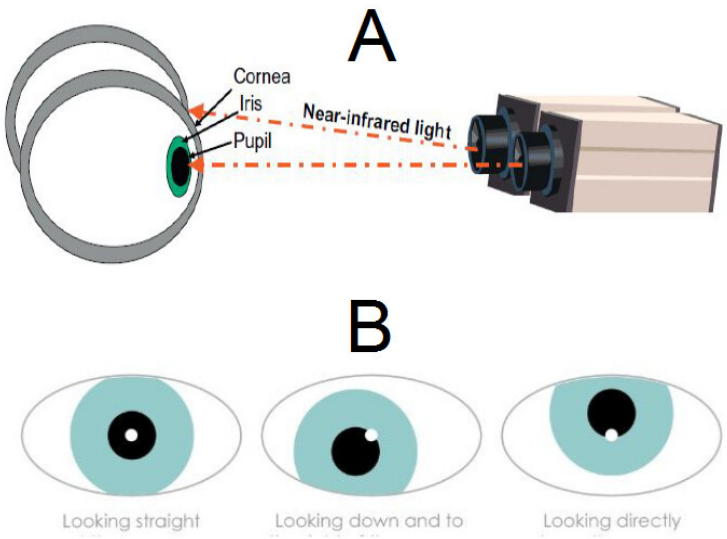

Figure 5: Eye tracking explanatory visuals

2.4 Self-driving wheelchairs

For some groups of users, even relatively simple inputs can be challenging to operate. Especially during driving, the user may have to continuously repeat these inputs. E.g. a certain eye-tracking wheelchair may require the user to always look at a screen, or always look in the direction they want to go. Applying self-driving features to the wheelchair can minimize the needed user input while still getting the user to the desired destination. Self-driving algorithms can do all the heavy lifting using path-finding and obstacle detection, while the user only needs to roughly indicate where he wants to go. Various research regarding self-driving wheelchairs has been done already.

A recent implementation can be found in Schiphol, which ran a trial of a self-driving wheelchair that can take passengers to where they are going, without the need for human input [17]. The wheelchair, shown in figure 6, can only follow fixed, preset routes, and will stop when it detects an obstacle. The passenger has to input the desired destination on a touchscreen. An emergency button is present in case the wheelchair malfunctions. The wheelchair cannot deviate from its programmed path and thus when it detects dynamic obstacles, it has to alert bystanders to get out of its way. While this implementation may be great for an airport, for the case of this project, the wheelchair needs to be “smarter”. The user should have more control over the destinations and the path it is taking, and the user input should be made more accessible as well.

Figure 6: Self driving wheelchair in Schiphol

Hye-Yeon et al. developed an autonomous self driving wheelchair for the physically weak [18]. They implemented SLAM in ROS with three Lidar sensors to create a map of the environment, and then generate a path from the current wheelchair position to the target position. The performance of the proposed solution was evaluated by comparing the path and driving behavior of the self-driving algorithm with real human input, and it was found that the applied methods were effective in driving from A to B. However, they also state that they did not implement many safety features. The target destinations also have to be typed into a chatbot, which is not convenient for an individual who can only use their eyes. Their prototype can be seen in figure 7 below.

Figure 7: Self driving wheelchair by Alkhatib et al.

Another autonomous wheelchair was developed by Alkhatib et al. [19]. It uses so-called April Tags, which look like QR-codes. These tags are sparsely stickered onto the walls around a building. The wheelchair is equipped with a simple camera. When the camera sees a tag, it can determine its current position and orientation based on the pattern, position and orientation of the tag in the camera frame. It combines this with simple infrared obstacle avoidance to be able to autonomously drive from A to B.

While there are various other autonomous wheelchairs being developed, there does not yet exist a robust, commercially available, eye-driven solution including all the safety precautions needed for the wheelchair to be operable in daily life.

2.5 Environment mapping techniques

In order to make a wheelchair “self-driving”, it needs to be equipped with the right sensors and software to make the equipment aware of its environment. Having a vulnerable person in a self-driving wheelchair comes with a lot of safety considerations, and thus the wheelchair needs to be smart enough to cope with different terrains, environments and weather-and lighting conditions, as well as the ability to react to unexpected situations that may occur. In order to achieve this, the wheelchair can be equipped with various sensors to map the surroundings and detect obstacles.

2.5.1 SLAM

Simultaneous Localization and Mapping (SLAM) is the term used for mapping an unknown environment while simultaneously estimating the robot’s position within that environment. SLAM allows a robot to build a 2d or 3d map of its surroundings by combining multiple sensor measurements and its own odometry information to iteratively estimate its current pose. SLAM is often the preferred method for indoor localization, as it can provide more accuracy and granularity than other localization methods like GPS. It can also function in dynamically changing environments, such as crowded places, where pre-constructed maps would be insufficient. [20] An example of a map created with SLAM can be seen in figure 8.

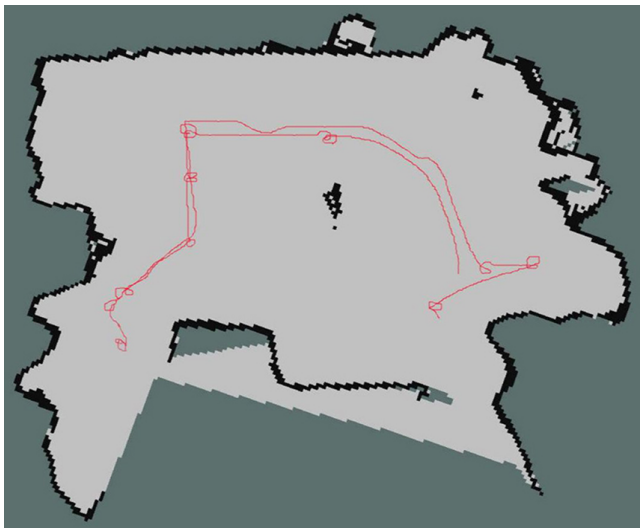

Figure 8: Example of a map created with SLAM

2.5.2 Sensors used in SLAM

In order to be able to run SLAM, the robot needs to be equipped with various sensors. The choice of sensors depends on the specific requirements, available

technologies and environmental conditions. In the most basic implementation, a Lidar and odometry sensor are used.

A LiDAR(Light Detection and Ranging) sensor emits laser beams and measures the time it takes for the laser pulse to return. Often this laser is mounted on a rotating element which allows it to take multiple readings per rotation to create a real-time 2d scan. Some more advanced LiDAR sensors can also create 3d scans. LiDAR sensors are used in many robotic applications, like self-driving cars, and can work both indoor and outdoor.

An odometry sensor provides an estimate of the robots pose based on the rotations of the wheels. Given the wheel mounting positions and the wheel radius, it's possible to calculate the relative path the robot is taking from its starting position by counting how many times each wheel has rotated. Wheel encoders are placed onto the wheels to be able to measure this. Odometry alone is, however, prone to drift over time. This can be due to inaccuracies in the wheel rotation measurements, slight inaccuracies in the static robot parameters, softening of the tires and slipping of the tires. Inertial Measurement Units (IMU's) are commonly used in conjunction with odometry sensors to counter this drift and obtain a more robust estimate of the robots pose.

In addition to LiDAR and odometry, other sensors like distance sensors, 2d- and 3d cameras, bumper sensors and GPS to obtain a better localization estimate and create more accurate maps with SLAM. [20]

2.5.3 SLAM software

To be able to process this sensor data and turn it into a usable localization and mapping, the right software has to be used. There are open source implementations available, almost all of which are on ROS.

The Robot Operating System (ROS) provides a set of tools, libraries and fixed protocols for building robotic systems. Many robots in development use ROS as its various open source libraries make it easy to prototype stuff, as well as make high-end, robust, low-latency implementations. It can work with almost any sensor and actuator, as long as it provides or accepts the right message formats. Code in ROS is most commonly written in C++ or Python. ROS also provides a standard communication protocol based on a publish-subscribe messaging model. This communication layer facilitates data exchange and coordination between different nodes within the ROS system. Each node can be a different sensor, actuator or piece of code, that can communicate with each other over the local network.

For SLAM, the most used open source libraries on ROS are GMapping, Hector Mapping and Google Cartographer. Each of which work in slightly different ways and their performance depends on the used hardware components and the environment in which the robot is deployed.

2.6 Control, Safety and liability

The development and deployment of a self-driving wheelchair raises important safety, liability and ethical considerations. How do we design the system in such a way that it runs stable and does not put the user in unnecessary dangerous situations? How should the system act when (parts of) the software or hardware malfunctions? In what ways can both the user and bystanders intervene when other software precautions fail? Who is liable in the case of physical damage to either the user or the surroundings? Questions like these are important to ask to make sure that the developed solution is both safe and does not cause any ethical concerns.

In order to address these questions, the Robotics Primer by Mataric [22] can be used as a guide. In this publication, different architectures related to control and safety are outlined.

2.6.1 Control

In order to ensure the safety of the developed system, eliminating potential hard- and software malfunctions is very important. This can be achieved by using high-end sensors, but mostly relies on how this data is processed and how the data flows through the different parts of the system. Since (wireless) communication is never perfectly reliable, it is safer to perform all needed processing on the robot itself. Sensors, actuators, and decisions need to interact in an effective way to get the robot to do its job. It is recommended to avoid controlling all of those elements from a single centralized program, and instead split them up into different blocks that are each responsible for a separate part of the system.

Control architectures provide guiding principles for designing programs and algorithms. The different architectures can differ substantially in how they handle time, modularity and representation, and the choice of architecture depends on the exact requirements of the solution, including but not limited to, the robustness to noise, static/dynamic environment, available sensor data, speed requirement, and the ability to look back, predict or learn.

The main robot control architectures are *deliberative*, *reactive*, *hybrid*, and *behavior-based*. In a *deliberative* architecture, every module (part) of the robot performs its task in sequence, where the output of one module is an input to the next. First they sense, then they plan, then they act. Things happen one at a time, and if one module fails, the whole system does. In *reactive* control, things happen at the same time, with multiple modules running simultaneously. There is a direct mapping between sensors and actuators; If sensor1, do actuator1. The whole system runs on a set of predetermined rules that act like reflexes, which makes it fast. In hybrid control, both deliberative and reactive control are combined.

The best known architecture for reactive control is Subsumption Architecture. The basic idea behind it is to build systems incrementally, from the simple parts to the more complex, and reuse already existing components as much as possible. They consist of a collection of modules or layers, each of which achieves a task. All of these modules work in parallel. When developing a robot using this architecture, we start at layer 0, e.g. move-around. Once this works, we add layer 1, avoid-obstacles, which works together with layer 0 to move around without bumping into obstacles. All layers can work together to achieve a certain task. Each layer can run independently, and when one layer fails, other layers can keep running unaffected.

An important aspect of the subsumption architecture is the ability for higher layers to temporarily disable one or more of those below them. When avoid-obstacle detects a too close by obstacle, it can disable move-around to stop the robot from moving. This allows for the implementation of layers with different priorities, where for example an emergency stop button is able to override the whole system.

A fully reactive control has its limitations, however, since it does not use any memory, there is no learning, and it does not use any internal models/representations of the world, which is something that may be required for a self driving wheelchair. That is why a hybrid control can be used, which combines deliberative and reactive control. Hybrid controllers can be made more robust and smarter than a reactive control. A hybrid system typically consists of three components; a reactive layer, a planner, and a middle layer that links the two components together. This is visualized in figure 9.

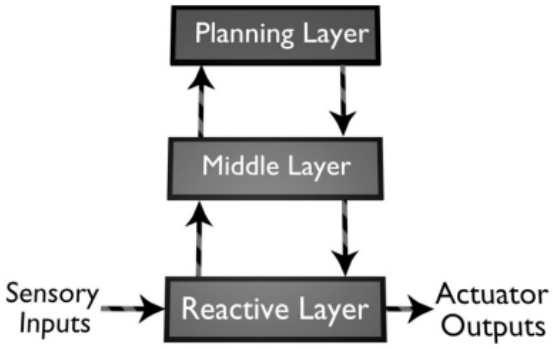

Figure 9: A possible hybrid controller design

For a self-driving wheelchair, the system needs to be able to quickly respond to unexpected obstacles and emergency stops, for which it needs a robust reactive controller. For determining a path for the wheelchair to travel, the system needs to use a deliberative controller to create an internal map of the perceived world. This makes hybrid control a suitable controller for a self driving wheelchair.

Designing the middle layer can be complex, as this layer is responsible for making sure that the two systems are brought together in a way that results in

consistent, timely and robust behavior. For example; how should the path planner respond when the reactive layer detects a dynamic obstacle that is in its way for longer than 10 seconds?

Instead of a hybrid controller, a behavior based controller can also be used. A behavioral controller is similar to a reactive controller, but allows for learning and representation as well. For a self-driving wheelchair, however, building a good internal map of the surrounding environment is essential, and with recent developments, map building and path planning can be done in close to real time, which means that the solution would not benefit much from switching to a slightly faster behavior based system.

Any developed system is never 100% reliable. Robots can produce unexpected behavior, also known as *emergent behavior*. Especially in dynamic environments, like the ones that a wheelchair may be used in, it is important to understand what emergent behaviors may occur, and how to deal with these in a safe way. The wheelchair may exhibit different behaviors than what it is designed for. Some of these behaviors may be features; an obstacle avoidance algorithm may cause the wheelchair to follow walls better, while other behaviors are not desirable; too strict obstacle avoidance can cause the robot to get stuck in a loop in certain situations.

2.6.2 Safety

While having a robust, reliable and responsive controller will make the solution a lot safer, it still depends on the right sensory data and reliable actuators in order for the whole system to be safe. A self-driving wheelchair cannot be completely safe if it is unable to detect tables, ledges, overhangs and cars, things which may be tricky or impossible for a single 2d lidar to detect. And even with the right sensor setup, no sensor is 100% reliable, and thus various emergency stops have to be implemented for the other 0.1%.

The sensor choice depends on the environment in which the wheelchair is used. If the environment contains tables with narrow legs or low overhangs, a 3D depth camera can be used to map those obstacles better. In order to prevent the wheelchair from driving off stairs, downward facing distance sensors around the wheelchair can be used to stop the wheelchair when it gets close to a ledge. When it bumps into something, a bumper sensor can be used to stop the robot. Furthermore, low acceleration and low maximum speeds can give the wheelchair more time to process data and reduce the braking distance.

Furthermore, both the user and caregiver should be trained in advance on how to control the wheelchair, and also how to identify and deal with potential unsafe situations.

Still, some situations can occur where despite all the precautions, the system puts the user in a dangerous situation. In this case, either the user or a bystander

needs to be able to intervene. In case of the user, this can be done by big on-screen emergency buttons where the user can look at. Ideally, however, if the user is able to control some other part of their body faster and more reliably, this should be used as an emergency input instead. Bystanders/caregivers need a way to intervene when the user is unable to. This is usually done with one or more big buttons on the robot, which can be pressed to shut down the full system. Caregivers can also be given a remote emergency button so they can intervene more rapidly. Emergency stops should always be implemented at the lowest possible level of the control architecture, and should override all other layers.

When an emergency stop gets triggered, whether from a manual intervention or from a dangerous situation detected by the software, there needs to be a way to re-activate the system as well. This could be done by the user, but it would be safer for a caregiver to assess the situation, make sure it is safe, and restart the system.

2.6.3 Liability

With any product, especially products in the medical (aids) industry, important liability concerns may be raised. Who is responsible if the wheelchair drives off and falls down stairs? Is it the manufacturer for not making a product that is able to detect 100% of all stairs, 100% of the time? Or is it the user who should have driven more carefully around the stairs and should have intervened when they saw the wheelchair drive toward the stairs? Do things change when the manufacturer advised beforehand to not use the wheelchair in spaces with exposed stairs?

Product liability law holds manufacturers responsible for accidents, injuries or damages caused by defects or malfunctions in the technology. No manufacturer intends to harm its users, and manufacturers will make sure their products are safe to use by both careful considerations during the development phase, but also with rigorous testing. Regulations and guidelines are established by governments to which they need to adhere to as well.

2.6.4 Ethical considerations

A self driving wheelchair can also cause ethical concerns. What if the wheelchair decides to go to a destination where the user does not want to go? It is crucial that the user remains in control all the time. But if the user has too much control, the wheelchair may pose risks to others. There needs to be a well considered balance between full autonomy and user control.

Another thing to consider is privacy and data protection. In order to realize the self-driving features, sensors like lidar and cameras need to be used, which gather personal data that the user may not want to share. It is therefore crucial that this data is safeguarded, for example by making sure that all of the data processing is done locally on the machine itself, without any data leaving it and without the ability for the

manufacturer to access private data. Still, the user may agree for the manufacturer to collect anonymized data to improve the system, which would also have to be done in a safe way.

Since this project is part of a bigger Ability Tech initiative, concerns about ethics and liability can be solved by partnering with existing companies like RDG Companje or Louwman, which have a lot of experience with helpcare assistive aids and mobility solutions. It can be hard to find and resolve all ethical, safety and liability issues by ourselves, but by sharing and collecting insights from companies who operate in the same market, many of these issues can be addressed.

2.7 Stakeholder analysis

2.7.1 Defining the stakeholders

Various stakeholders can be identified in this project. These include developers, end users and other related parties. Here follows a more thorough overview.

- **End User:**
  Users with physical disabilities for which the wheelchair will be developed. This is the primary stakeholder and can provide valuable insights and feedback throughout the project through interviews and user tests.
- **Caregiver and/or family members:**
  As it is more challenging for the end user to communicate their exact needs, the input of caregivers can be helpful. The caregivers and family members have experience with caring for the end user, and their perspective can supply more information about their needs and can help think about possible solutions.
- **Government:**
  Who set up specific laws one needs to abide by, and can provide necessary funding for the project.
- **University:**
  Sets up specific time constraints and requirements for the project.
- **Maker spaces:**
  Work spaces like the Designlab and Fablab that provide equipment like 3D-printers, laser cutters, woodworking tools, etc.
- **Hardware suppliers:**
  Webshops and local stores where required components can be purchased. Need to monitor their stock and shipping durations. These can include webshops like Conrad and Tinytronics for electronics, but also local hardware stores like Karwei for materials and equipment.
- **Soft- and hardware engineers:**
  Developers who are working on the realization of the project, both in the hardware and software side. For this wheelchair project, this mostly consists of me, with occasional help of two other people. For further development in the scope of Ability Tech, possibly expertise from outside the university is needed. This can be R&D facilities or existing mobility and healthcare aids companies. (e.g. RDG Companje or Louwman)
- **Ethics Committee:**
  Board which needs to give approval to perform various interviews and user tests.
- **Medical departments**
  Departments that operate in and regulate products and services related to the medical (aids) sector. These can include medical insurance agencies, health

and social care assessors, and WMO (Medical Research Involving Human Subjects Act) consultants.

- **General public:**
  Since the wheelchair is being operated in a public environment, bystanders can also be considered stakeholders. The wheelchair may need to navigate through busy areas, and may need to indicate to people where it intends to move and possibly make known that someone is blocking its path. Furthermore, the ability move will extend the possibility to socialize for the end user.

2.7.2 Stakeholder matrix

All defined stakeholders have been placed in a stakeholder matrix to get a better overview of the involvement of each stakeholder. It's important to inform and monitor the stakeholders before, during and after the project, to make sure that the requirements are up-to-date and there are no unforeseen problems arising that could delay or otherwise negatively impact the process of the project.

For this project, a stake-holder matrix can be created as shown in figure 10:

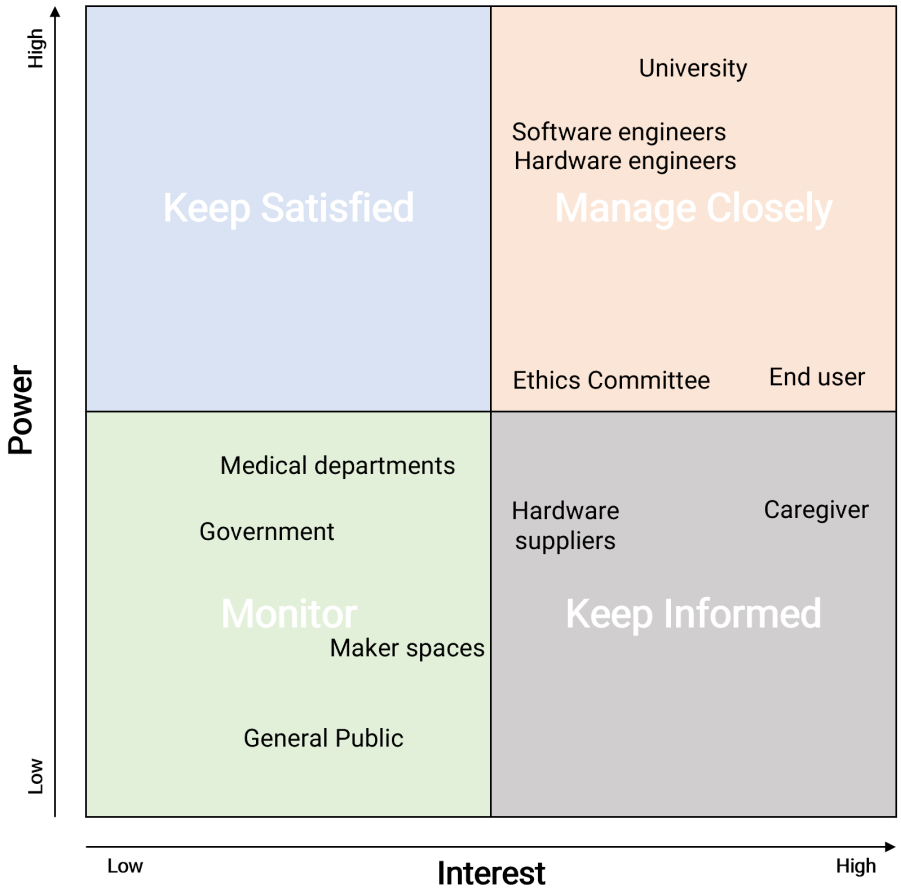

Figure 10: Stakeholder matrix

2.8 Requirements

To be able to come up with a concise set of requirements, an interview was carried out with both the caregiver and the end user. The interview questions can be found in Appendix A. The main goal of the interview was to create a clear overview of the exact needs and desires of the user, as well as getting some practical information that may be useful during the development of the project.

2.8.1 End user profile

Based on this interview, we can come up with a detailed profile of the end user: The end user for which the wheelchair will be developed is Andrei Fokkink, a 26 year old male. Due to birth complications, Andrei is unable to voluntarily control any of his muscles except for his eyes. He does have a small degree of control over his neck movement, but not consistent enough to be usable. He is also spastic, which leads to random unvoluntary muscle movements from time to time. Andrei is wheelchair bound and relies on caregivers to move around and perform daily tasks like eating.

Despite this, his cognitive functions are mostly unaffected and his senses like seeing, hearing, feeling, tasting and smelling are working normally as well. He uses simple eye movements (looking away/at something to indicate a no or yes) to communicate, and also uses a Tobii eye-tracking tablet to construct full words or sentences. This allows him to participate in social interactions and make known his needs.

In his free time, he likes to go scouting, make music and make art. Using his Tobii tablet, he is able to control a pen tool in a paint-like program, in which he draws abstract art, which he also sells. He also uses the tablet to write emails, send text messages, and to browse the web. The accuracy of the eye-tracking and the focus of his eyes is enough to be able to perform these tasks, although misclicks do happen and clicking on small on-screen elements can be challenging. Using the eye-tracking for a prolonged period of time can also lead to discomfort.

Ideally, the wheelchair should be usable on regularly visited locations at the least, being two houses and the day care. Andrei said that he would also like to be able to drive around outside, for example on the streets or in the park. He expects to actively drive around with the wheelchair for around two hours a day, but that may change when he has more freedom of moving around.

When asked why Andrei wants a self-driving wheelchair, he indicated that he mainly wants to be able to move from A to B whenever he wants, without needing to wait for a caregiver, and to “unburden” the caregiver by not needing someone to drive him around. He wants to become more independent and be able to more actively be a part of the society.

2.8.2 Project requirements

Based on the interviews and technical possibilities, the following list of project requirements can be established, according to the MoSCoW priority categories [16]:

MUST HAVE

1. The end user should be able to move from A to B completely independent of caregivers in a spacious, pre-known environment .
2. The wheelchair should be safe to use and should not expose the user to any unnecessary risks. This can be achieved with the use of simple obstacle detection, as well as accurately being able to detect user input and possibly adding emergency stops. Together with the ethics committee, the risks associated with the user tests should be considered.
3. A way for the user to make known his desired destination should be created, most likely using an interface on an eye-tracking tablet.

SHOULD HAVE

4. A way to map the environment to be able to implement more advanced obstacle detection
5. A low-effort GUI which allows the user to make known his desired destination without needing to focus on the screen 100% of the time while driving.
6. The ability to navigate and drive around in new environments

COULD HAVE

7. Long-lasting battery so the user can satisfy his driving needs for a full day (~2 hours of driving)
8. Robust hard- and software that are not specific for the used wheelchair frame, but is able to be integrated into a variety of existing wheelchairs with low effort.
9. Full path-finding using SLAM, where the user only needs to click on a point on a map, after which the wheelchair fully autonomously drives to the destination.
10. The ability to detect dynamic obstacles (e.g. humans) and be able to notify them to move out of the way in case they are blocking the path.

WON’T HAVE

11. The ability to drive around outside. Given that there are no walls outside to use as a point of reference, the requirement for sensors to function in different weather conditions, and the safety considerations needed with respect to traffic situations, making the wheelchair be fully operable outside will most likely be out of scope for this project. However, driving outside is a wish of the client, and wheelchair users in general, so during development, certain hard- and software decisions will be made so that these could work outside in future iterations.

3. Ideation

In order to meet the requirements, a basic ideation about what the final setup may look like can be established.

A basic high level diagram of the intended solution can be found in figure 11. First, a map is built using SLAM. The user can see the map and the current position and orientation of the wheelchair in that map on his screen. He can input a target destination, after which a path is created. This path is then shown to the user for confirmation. After confirmation, the wheelchair drives to the destination using the generated path, whilst continuously updating its path based on changes in the environment. During the driving phase, the user is able to intervene when needed.

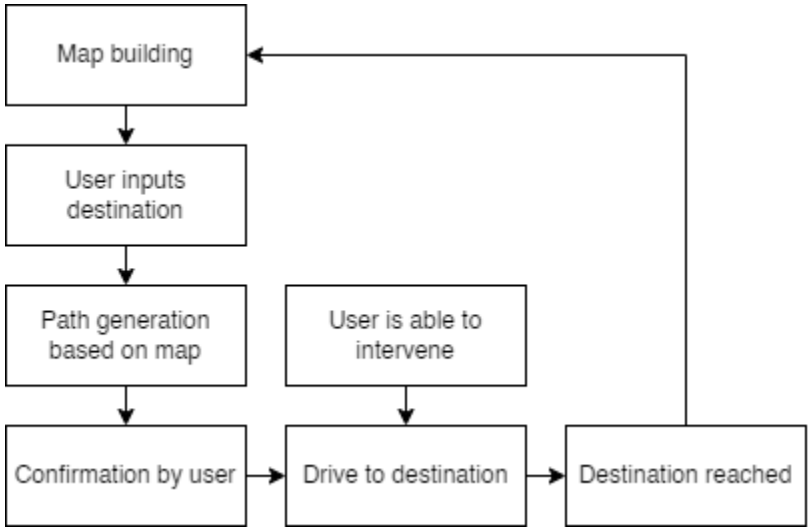

Figure 11: Basic block diagram of the system

A possible way of displaying this information to the user can be seen in a mock-up user interface in figure 12:

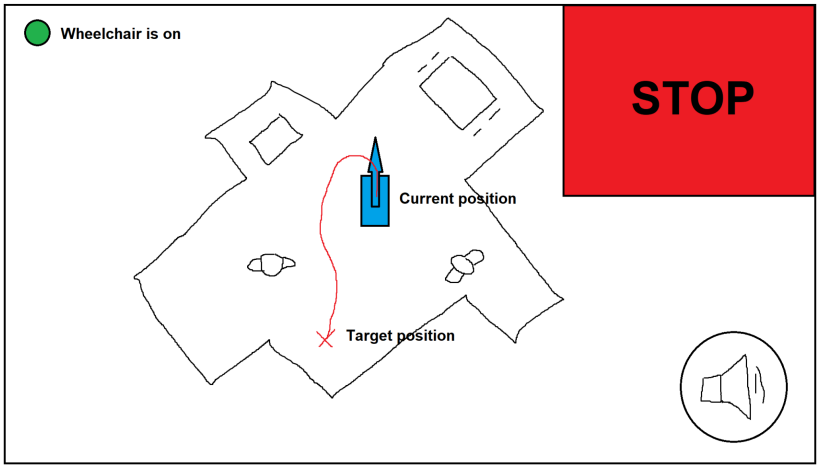

Figure 12: Sketch of a possible user interface

4. Method/setup

In order to answer the research questions stated in Chapter 1.2, and to translate the requirements of Chapter 2.8, one or more prototypes will be developed. These prototypes will be iteratively improved and tested, adding more layers of complexity after each iteration. The first prototype(s) will consist of simple, non-autonomous controls, to get a feel of the platform and the abilities of the user. After this, the self-driving features will be implemented.

4.1 Wheelchair setup

4.1.1 Wheelchair base

For the prototype, an old wheelchair base was in possession by my supervisor and readily available to be used. Ideally, a more modern setup would be used, since they include built-in batteries and often useful built-in sensors. However the decision was made to go with the current base since it would cost more time/money to obtain a modern base, and would possibly require more complicated fiddling with protocols to be able to tap into the controls and sensor data.

The used wheelchair base is made by Dynamic and is a differential drive setup, with two swivel wheels at the front. Originally, the wheelchair can only be controlled using a joystick, but using digital potentiometers the joystick input is simulated, which allows the base to be controlled programmatically. Two large 23V SEM motors power the two main wheels, which have solenoid brakes at the back side of the motor shaft for mechanical braking. A control box is attached to the motor controller, which houses a joystick, an on/off button and a rf-receiver. A separate remote controller with a Wii Nunchuk is used to be able to easily control the base from a distance.

The base is powered by a modified uninterruptible power supply (UPS). The electronics in the UPS are used for charging and protecting the batteries, while the two 12V lead batteries are connected in series to the motors and the rest of the system for a 24V power supply. The batteries each have a capacity of around 20Ah, which should be enough for a few hours of driving. The wheelchair base, along with its controllers, can be seen in figure 13.

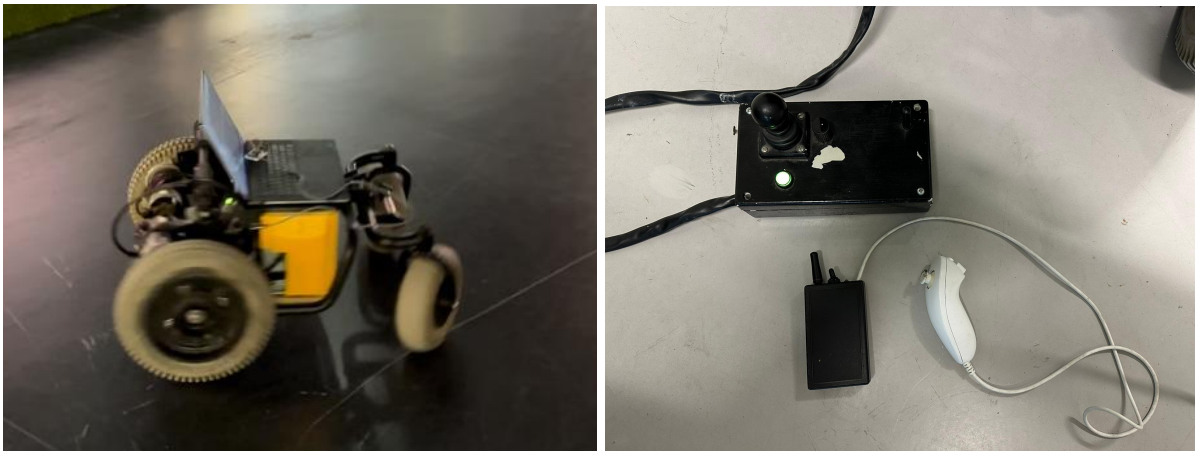

Figure 13: Wheelchair base with controls

4.2 Odometry

4.2.1 Encoders

In order to get reliable position data for SLAM, and for being able to apply PID control, having good odometry information is a must. The standard way of adding odometry to a differential drive robot is to add two encoder disc sensors to the back of the motors. These encoder discs consist of many small holes or small magnets at consistent angular intervals, and are combined with sensors. The sensor counts the number of ticks to determine how fast and in which direction the motor is turning.

Modern electric wheelchairs have built-in encoders, of which the values can be obtained by tapping into the communication protocol used by that wheelchair. Depending on the age and brand of the wheelchair, different protocols may be used. These protocols are often not open source, and may require a direct line of communication with the wheelchair manufacturer to get the exact specifications of the protocol.

The wheelchair base in the used setup is relatively old, and does not include any built-in encoders. It was also not feasible to use a conventional encoder, since the motor shaft on the back of the motor was occupied by the brakes. Furthermore, there was also not much room to put an encoder disc directly on the wheel-axis behind the wheel, especially while needing to be sturdy and weather-proof. Therefore a different odometry setup had to be designed. Designing a suitable odometry setup will not only be useful for this project, but having a setup that can fit on any type of wheelchair may also be useful for future implementations where direct odometry information from the wheelchair may not be available.

4.2.2 Encoder design

The two main types of encoders are Optical and Magnetic. In the case of optical encoders, encoder discs are used with many small holes along the edge, through which an LED shines through. This is not usable in our setup since there is no room to put encoder discs. Therefore, the decision was made to go with a magnetic encoder.

By placing two strong magnets on the wheel at opposite orientations, a constantly changing magnetic field can be generated as the wheel turns. A hall-effect sensor can be used to measure the absolute value of the magnetic field. The hall sensor can be placed on a fixed point on the wheelchair base, close to the magnets. The resulting output will resemble a periodical, sine-like wave. Adding a second sensor at a 90 degree angle with regards to the first will result in a cosine-like wave. **The two hall sensors combined will always yield a unique pair of measurement values for each absolute angle of the wheel**.

An illustration of the setup can be seen in figure 14

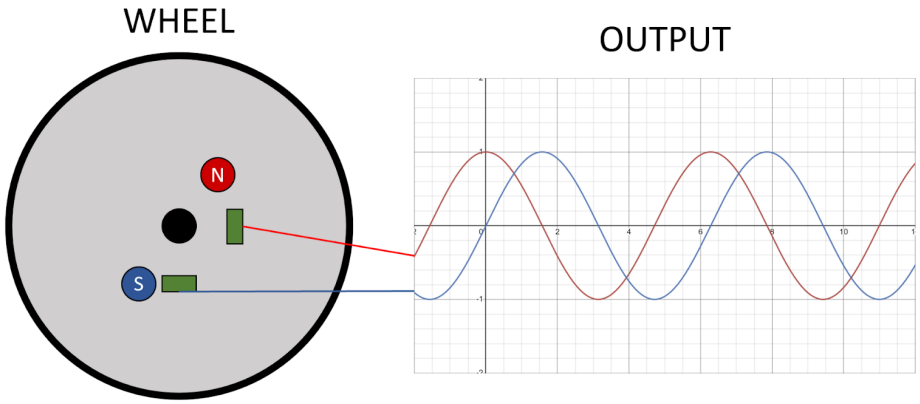

Figure 14: Odometry setup using two magnets and two hall sensors

This setup should work for any wheelchair, or in fact almost anything with wheels, as it only requires to have the ability to mount two magnets to the wheels, and two hall sensors to the base. Magnets can be mounted using screws, clips, glue or tape, as long as they are fixed in place with minimal chance of displacement over time.

4.2.3 Encoder setup

The magnets should have a sufficiently strong magnetic field to minimize the effect that other external magnetic fields(noise) have on the measurements. Therefore the decision was made to go with N42 magnets, which are strong and relatively cheap. After some initial tests, it was found that one magnet on each side wasn't strong enough, so two magnets were placed on each side. Two hall sensors were

placed right along the wheel axis, at approximately a 90 degree offset. The placement of the hall sensors and magnets can be seen in figure 15 below.

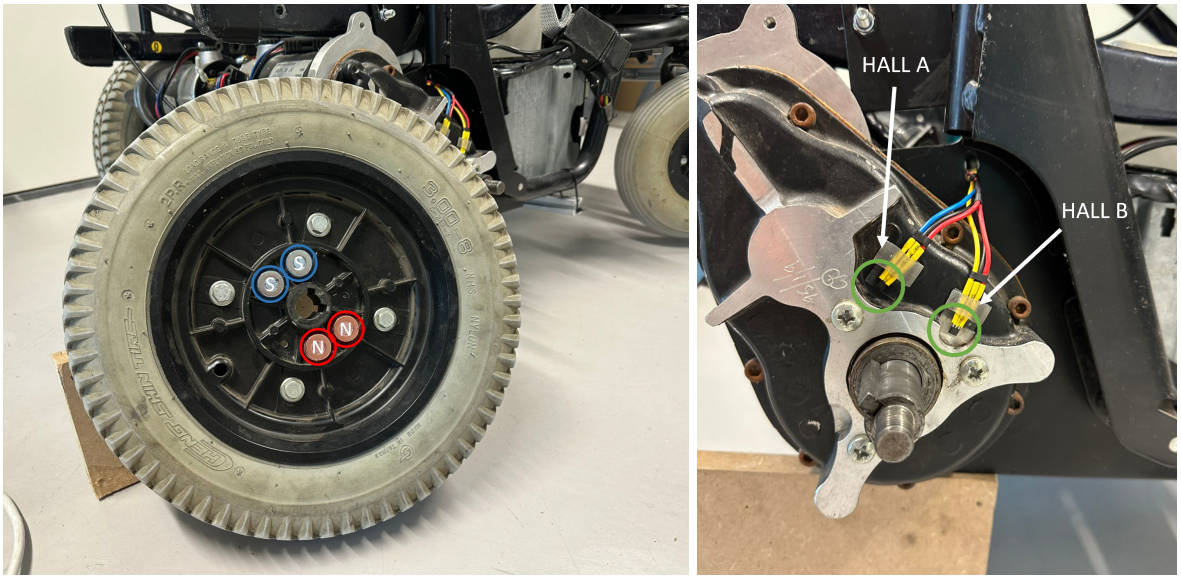

Figure 15: Placement of the magnets and hall sensors on the wheelchair.

The sensors are hooked up to a Teensy 4.0, which is reading the values and printing them out. Teensy 4.0 was used because of the ability to read analog values in a higher resolution (up to 16 bits), and because it is faster than a standard Arduino, so more measurements can be done and processed per second. Plotting both values as the wheel turns at a consistent speed yields the graph shown in figure 16. It can be seen that both sensors yield a similarly shaped periodic signal with an offset. At no two points in the same period are A and B the same, e.g. every combination of A and B measurements yields a unique angle. The signal has a “bump” at the top and bottom of the curves because two magnets were used on either side. This should however not impact the accuracy of the angle extraction as each A/B pair of values still yields a unique angle.

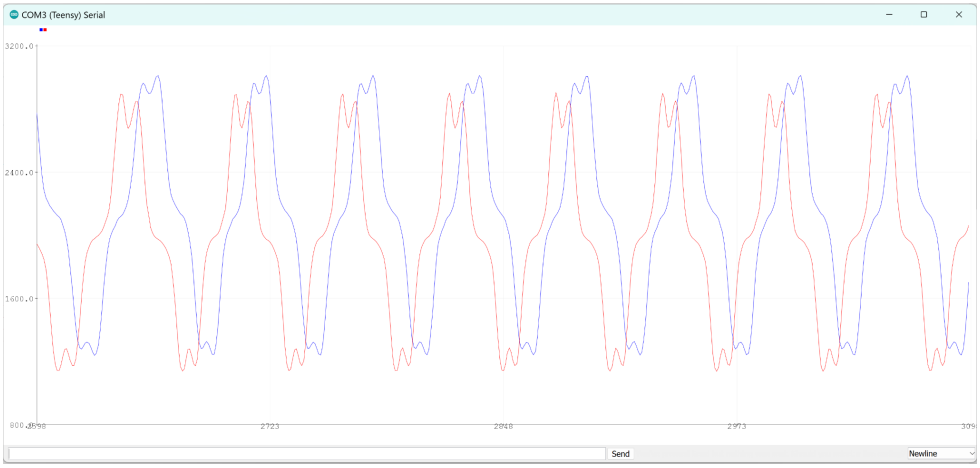

Figure 16: Hall sensor output

In order to turn the raw hall sensor outputs into absolute angles, a calibration method was implemented. While turning the wheel at a constant speed, the operator can push a button connected to the Teensy. This will make it start recording the readings of both sensors until it has completed a full rotation (at this point the pair of A, B values is (almost) the same as when it started recording). A buzzer indicates when this calibration is done. Both sensor values are then saved to an array. Since the wheel is turning at a constant speed, each reading is a fixed angle apart. E.g. if 1000 readings were made, each reading is 360/1000 = 0.36 degrees. These arrays are saved into the internal persistent memory of the Teensy.

To compute the current absolute angle, a simple nearest-neighbor algorithm is used, which takes the current sensor readings, and computes which calibrated reading pair is closest to it. This results in an array index, which can be converted to an angle using *angle=index/total_calibration_readings*360*. A moving average filter is applied to filter out noise. Since a reasonably powerful Teensy 4.0 is used, readings are taken 200 times per second, with a 10 reading moving average window. This results in accurate angle readings with a sufficient resolution, and since the computed angles are absolute, there is no drift over time, which could be the case with normal encoders. The resulting angles are integrated over time to get the distance traveled for each wheel, which are then published in ros.

4.2.4 IMU

Just using wheel encoders for odometry can lead to inaccurate results in some situations. If the wheelchair drives on a slippery surface and the tires slip, or the wheelchair is stationary stuck at a wall while the wheels are turning, the wheel encoders will accumulate a lot of errors. Even in almost perfect conditions, a slight change in the tire pressure (smaller wheel diameter), or inaccuracies in the calibration can result in large deviations over time. Therefore, other odometry sensors can be used to make it more robust.

A commonly used counterpart to the encoders is the IMU. An Inertial Measurement Unit(IMU) measures triaxial acceleration, triaxial angular velocity, and sometimes orientation using earth's magnetic field. If the sensor is moved forward one meter, this results in a certain change in acceleration, which can be integrated to get distance. IMU’s, especially cheap ones, do drift a lot over time, so cannot reliably be used on their own as odometry sources. But in combination with wheel encoders, they can greatly improve the accuracy of the odometry as a whole.

For this project, the Xsens MTI-1 was used, as it is a high-end IMU sensor with built-in magnetometers.

4.3 Lidar

During the first phase of the development, the SICK TIM571 was used. This SICK scanner was salvaged from a different robot that was present at the faculty. The scanner is high-end with a range of 25 meters and an angular resolution of 0.33 degrees. While it worked really well, the sensor is also relatively expensive, with a new price of around $2,000. Given that we would like to make the setup production-proof, the decision was made to switch to a cheaper alternative.

The used lidar sensor is the RPLIDAR S2, which is a 360 degree laser scanner with a range of 30 meters and an angular resolution of 0.12 degrees. It has better specifications than the SICK scanner, while also being considerably cheaper at $400. The lidar is connected to the LattePanda using an USB connection, and is positioned at the front of the wheelchair base. The lidar, along with its lidar feed, is found in figure 17.

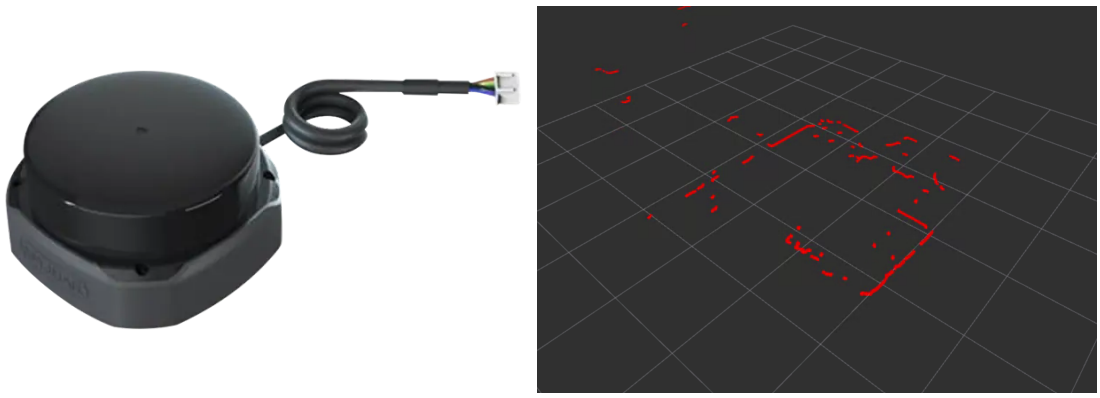

Figure 17: RPlidar S2 and the corresponding lidar feed

4.4 Other hardware components

4.4.1 Processor unit

For the main processing unit, the LattePanda 3 Delta was used. A Raspberry Pi 4 would in theory also work just fine, but to make development more efficient, a more powerful board was desired. During development, code may not be very optimized, and (3d) visualizations can be very helpful for understanding why things do and don't work. Having a more powerful board means that more rich and informative visualizations can be made, and prototypes can be built faster. The Raspberry Pi is also hard to obtain lately as it is often sold out everywhere.

The LattePanda 3 Delta comes with Windows pre-installed, but is manually flashed with Ubuntu 22.04, since Linux is better for robotics and ROS. It comes with three USB 3.0 ports, as well as an Ethernet and HDMI port. A HDMI dummy plug is inserted into the HDMI port in order to make it possible to use a remote desktop

viewer to control it from a distance. The LattePanda has a built-in Arduino Leonardo board, of which the pins are directly exposed and ready to be used.

4.4.2 Camera

As the wheelchair user is unable to move their neck, they cannot look next to or behind them. While the Lidar scanner should be able to detect most obstacles, some small or low obstacles may be missed, in which case the user needs to be able to intervene. Therefore a good view of the user's surroundings is very important.

In order to facilitate this, a wide-angle fish-eye camera is placed about 1 meter above the wheelchair, pointing down. The camera feed is shown in figure 18. This allows the user to look around in an approximate 2 meter radius. In future versions, this can be replaced with an actual spherical 3D-camera, which allows a bigger viewing radius and doesn't require to be mounted very high above the wheelchair. An alternative setup would involve placing multiple cameras around the wheelchair and stitching the different camera feeds together, in which case the cameras can be mounted even lower, a setup often used in smart cars.

Figure 18: Top down camera view

4.4.3 Power

Since the wheel motors require 24V and a large current when driving, a suitable power supply had to be used. The wheelchair base already included a modified Uninterruptible Power Supply (UPS), with two 12V lead batteries inside. The PSU electronics are used for battery management and charging, while the two 12V batteries are connected in series to make 24V, and are connected to the rest of the

wheelchair. In between the batteries and the wheelchair, a BMS board is placed to protect against short circuits and for under- and overvoltage protection.

Since the LattePanda requires 12V, a separate 24V->12V 5A DC-DC converter by VictronEnergy is used to step down the voltage. The LattePanda has an on-board 12V->5V converter which is used to power the Teensy.

4.4.4 Safety components

In order to make the prototype safe enough to perform user tests with, some emergency stops had to be implemented. During the user tests, the researchers can override the control at any time. For an extra safety layer, it was desired to attach a bumper sensor to the wheelchair, in order to make it stop when it hits anything. However, after some online searching, no suitable bumper sensor could be found that could fit around the wheelchair.

Therefore, a custom bumper was designed, which can be found in figure 19, consisting of a long red wire, some bearings and two limit switches. When an obstacle hits the red wire, the wire bends, causing the limit switches placed near the handles to be pushed in. The plan was to add small 3d-printer bumper sensors around each of the bearings, in case the obstacle hits a corner first before hitting the wire, but due to time constraints we were unable to implement this. Despite being a low-end solution, it works decently well, although it requires more travel than a normal bumper sensor.

Besides functioning as a bumper sensor, it also provides a clear and easy way for bystanders to stop the wheelchair in potentially dangerous situations. By pulling the red wire, this also causes the switches to get activated, and thus functioning as an emergency stop. The red color of the wire communicates to other people what its purpose is, and it should be an intuitive way to intervene.

For lower obstacles, a more conventional bumper setup with extrusion profiles and limit switches was created at the front of the wheelchair base.

Figure 19: Red wire emergency sensor

4.4.5 System overview

A diagram with all main components of the setup can be seen in figure 20 below.

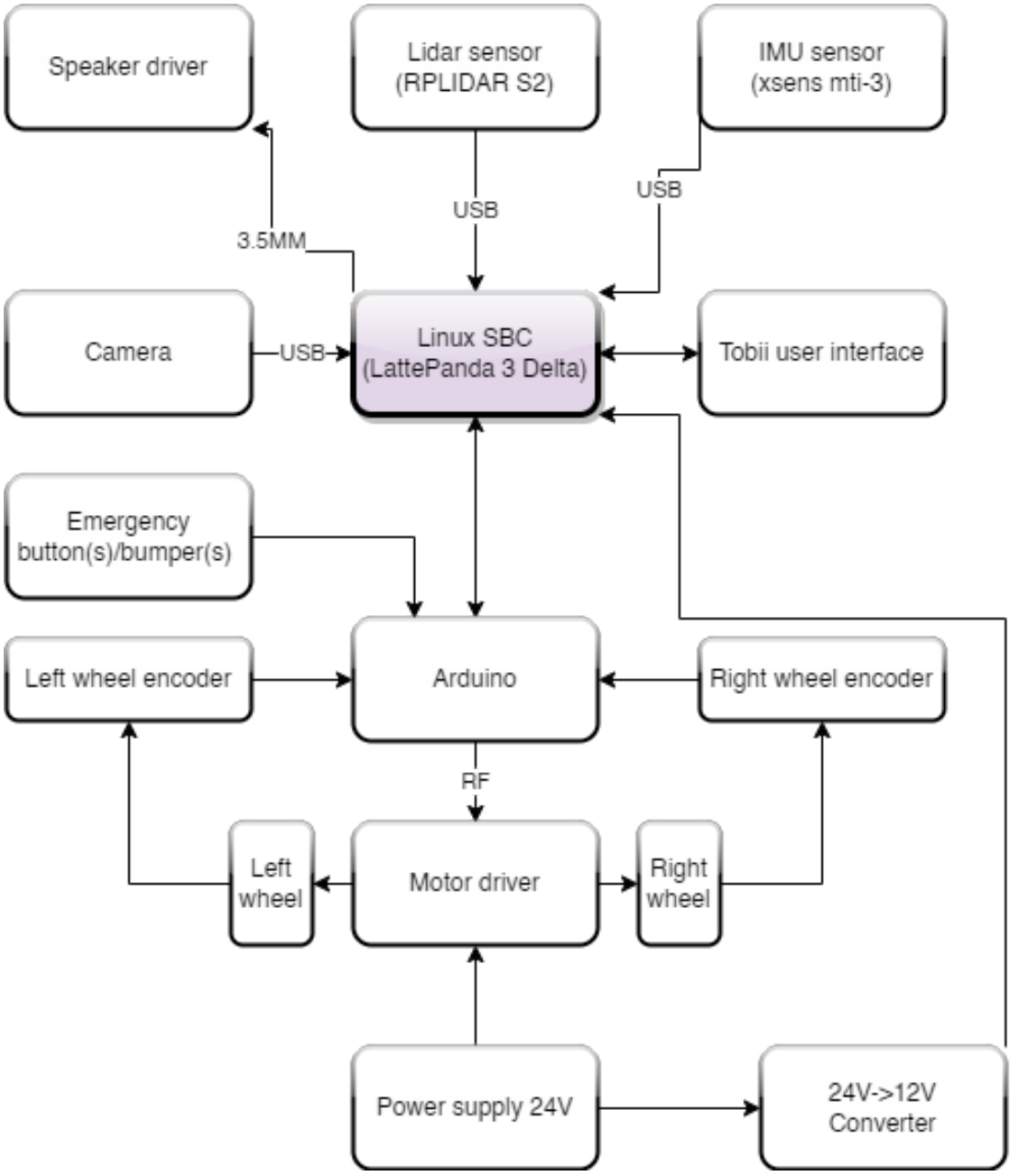

Figure 20: Block diagram of the hardware setup

A photo of the full wheelchair setup (apart from the red wire bumper), is given in figure 21:

Figure 21: Wheelchair prototype

4.5 Software

The software of the wheelchair is designed according to the hybrid control architecture discussed in chapter 2.6.1. During development, layers are iteratively implemented starting from simple low-level layers like odometry and joystick control, to more complex layers like SLAM. When one layer works, a new layer can be added. Different layers can be combined for different software configurations. The software layers used in the first and last user tests are almost the same, apart from the main layer that combines all the inputs. Every sensor runs independently in their own layer(ROS node), in parallel to all the other sensors. Each prototype has his own middle layer that processes the sensor data in different ways for different outputs.

The different layer hierarchies have also been implemented. Most of this happens in the main layer itself, which is responsible for collecting and combining the sensor data. It will stop the wheelchair if it detects certain sensor feeds are malfunctioning. It also has different internal hierarchies, for example when close obstacles are detected on the lidar feed, the whole wheelchair will stop. The emergency switches are fully in hardware and override all software, they are the lowest level layer.

For the software of the wheelchair, a ROS1 Noetic environment was used. While there are more recent environments based on ROS2, ROS1 was chosen because despite ROS2 being advertised as being fully backward compatible, the Xsens library only seemed to work with ROS1. Furthermore, the Arduino Leonardo built into the LattePanda could not be gotten to work with ROS2 as well. ROS is a widely used, open source platform, which makes development easier and also makes it easier for future work to be built on top of this software.

In figure 22 below, an overview of all the ROS nodes can be seen:

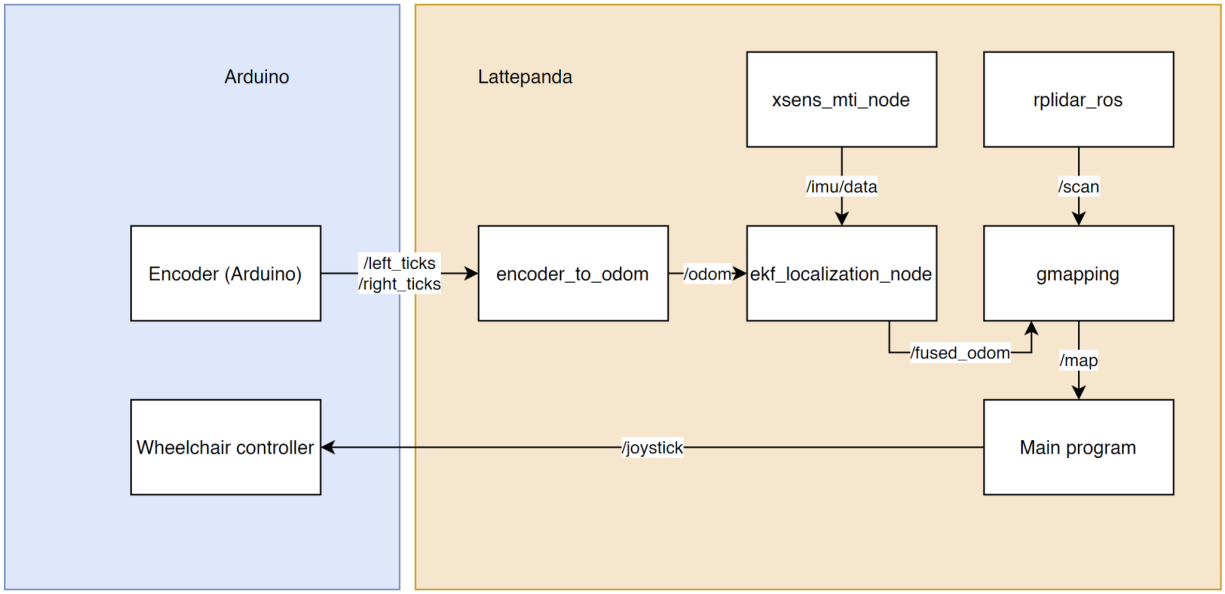

Figure 22: ROS nodes block diagram

4.5.1 Odometry

The Arduino node receives encoder ticks information from the teensy and publishes it to the LattePanda in *left_ticks* and *right_ticks* topics. These topics represent the number of encoder ticks that each wheel has turned since the system booted up. Because of the custom odometry system, this could be set to 100 ticks per degree. As an example, left_ticks could be -29456 and right_ticks could be 68123, meaning that the left wheel did almost a full rotation (294 degrees) backwards, while the right wheel did almost two rotations (681 degrees) forward.

In order for the Arduino to be able to publish ROS messages, the Rosserial library is used. This package allows devices to communicate over serial using the ROS protocol.

This data is then passed into a Python node that uses this tick data to compute odometry information. An odometry message consists of a pose and a twist. A pose is simply the current position and orientation of the robot, while the twist is the current velocity expressed in both its linear and angular parts. The pose is computed by taking the total rotations of each wheel and using math to calculate the current pose. The twist is calculated by taking the difference between the current last tick messages, and applying math to that. For this, open source code was used.

In order to make the odometry more robust, the Xsens IMU sensor was added as well. The *xsens_mti_driver package* was used to take data from the Xsens and publish it on the /imu/data topic. In order to fuse both data streams, the robot_localization package was used. This package can fuse an arbitrary number of sensors and use filters to combine them into a more reliable data stream. In this case, an *ekf_localization_node* was used to fuse the IMU and encoder data. The EKF node combines both datastreams by using an Extended Kalman filter. The needed covariance matrix was iteratively tweaked until a smooth fused odometry frame was obtained.

The resulting odometry output can be seen in figure 23 below. In this example, the wheelchair was driven around the RAM lab at the University of Twente. In the ideal case, the wheelchair would start and end in the same position. Even though fused odometry was used, slipping of the wheels still caused a noticeable drift. However, this should be accurate enough to be used in SLAM.

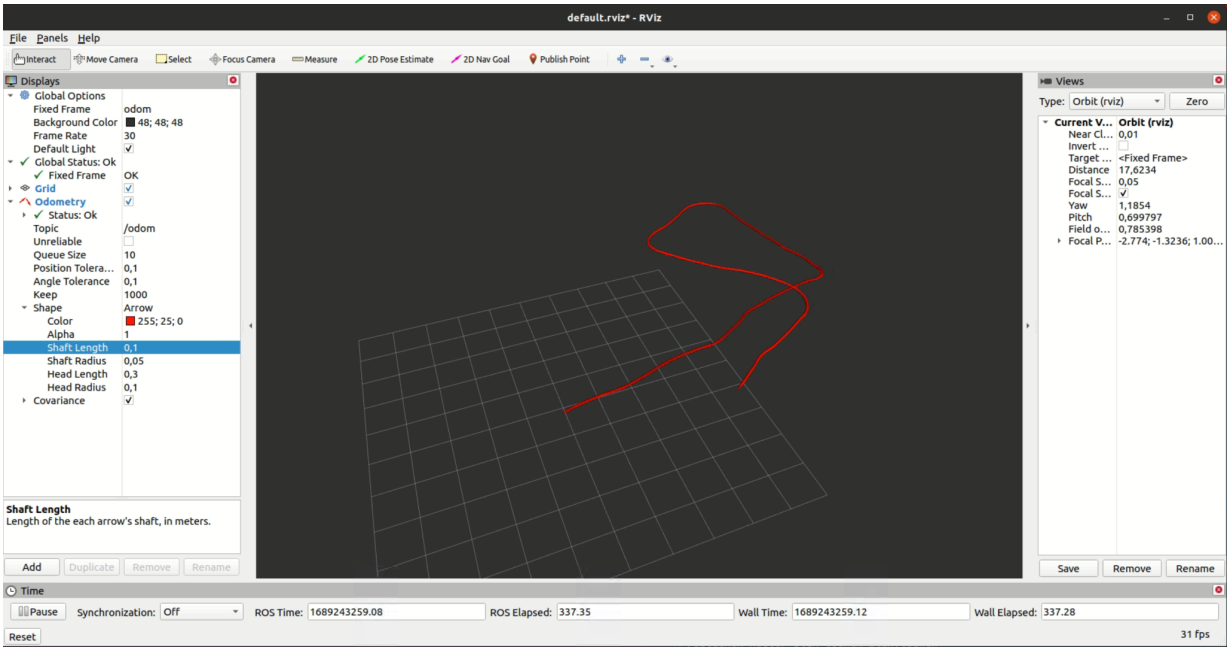

Figure 23, Odometry output in Rviz

4.5.2 Lidar

In order to be able to obtain data from the RPlidar S3, the *rplidar_ros* package was used. This package takes the raw lidar data from the RPlidar and transforms and publishes it as a ROS-compatible LaserScan message.

4.5.3 Gmapping

To be able to construct a map of the wheelchairs environment, the *gmapping* package was used. This is the most used SLAM package in ROS, with a lot of available documentation and example code, which makes it convenient to integrate. It subscribes to the /laserscan and /odometry topics and turns this into a 2d, grid-like map.

4.5.4 Main program

The resulting map, as well as the raw odometry and lidar data, is being published to a main node that combines these data streams in a useful way. This node is custom written in Python. This node converts the odometry and map information to a websocket-friendly format, which is sent to the GUI on the user's tablet, which runs in a browser. From the GUI, it either receives direct joystick control commands (e.g. 127,231), or a x,y target destination on the map.

In the latter case, it will first compute a path to that target destination. Since I was struggling with implementing full SLAM-based path finding, the decision was made to implement a more simple approach with just the odometry information. When

a target (x, y) destination is given, the program first computes the target heading. If its current heading has a too large deviation from the target heading, i.e. the wheelchair is facing the other other way from where it needs to move, the wheelchair will first rotate around its axis until it is more or less facing the right direction (less than a quarter pi radians offset). It then starts driving forward, but with a slight x velocity to correct its heading and steer itself towards the target point. It then keeps driving until it is less than 10 centimeters away from the target position. This works pretty well, although inaccuracies in the odometry information (no SLAM is used here), produces some drift from the intended target destination. This drift is mostly caused by the slipping wheels, but also by the fact that the swivel wheels at the front are not modeled and cause the wheelchair to move differently depending on the current position of the swivel wheels.

In both cases, before the velocity (joystick) input is passed on to the motors, the lidar feed is used for simple obstacle detection. It will take the intended velocity input, and translate and rotate the current lidar feed based on what it will look like in about 1-2 seconds if the current velocity input is passed on. This can be seen in figure X below. The white dots on the left are the current lidar points, while the gray dots are where they would be 1-2 seconds from now given the current joystick input (right). If more than 10 points are inside the rectangular perimeter of the wheelchair, it indicates that the wheelchair is about to bump into something. 10 was chosen to prevent triggering unwanted behavior due to noise, but at the same time still being able to detect narrow objects like table legs. If it is about to bump into something, instead of passing on the velocity commands, it will send a 0-velocity command instead (127,127), to make the wheelchair break. To keep moving, the user simply has to point the joystick in an unobstructed direction, or in case of the path finding, set a new target destination.

The resulting velocity command, either the direct joystick input or the output from the path finding, is then published to the Arduino, which sends it over RF to the motor controller.

4.6 User interfaces

4.6.1 UI architecture

Two versions of the UI were made, one written in Pygame, a visualization library for Python that runs on the Lattepanda, and P5js, a Javascript visualization library that runs in the user's browser. The reason for this is that pygame runs on the wheelchair itself, which makes prototyping more efficient as there does not need to be any data exchange between the wheelchair and the user's browser over a (private) network, and user inputs can be directly published over ROS. However, during user testing, the user would need to use a remote desktop application like Teamviewer to

access the Pygame application on the wheelchair, which adds noticeable latency especially when using the joystick.

Therefore it is better to run the GUI program on the user's Tobii tablet directly. To make it more convenient to test things, it was decided to go with a browser application since this will allow the GUI to run on any machine without needing to install any programs or dependencies. Since there was previous experience with P5js, this was chosen as the main programming library to develop this in. Lidar and camera data are converted into a websocket-friendly format and then sent to the client over a websocket running on the local wifi. The client then sends back the user input over the same websocket.

4.6.2 UI design

At first, a simple joystick GUI was made, with which the wheelchair can be controlled with either a mouse, or with the eye-tracker. The Tobii eye-tracker emulates a mouse pointer, so both inputs can be treated in the same way. This joystick is helpful for testing, as it allows for controlling the wheelchair from any device directly. The joystick area is made as big as possible to allow the user to be less precise in their eye movements. The wheelchair will only start driving when the user looks at the joystick area for longer than 0.5 seconds, to avoid triggering on quick glances. When the user looks away from the joystick area, the wheelchair stops, which means that the whole rest of the screen can be treated as an emergency stop. This should make it safe as the user only needs to look away if something goes wrong. The GUI can be seen in figure 24 below.

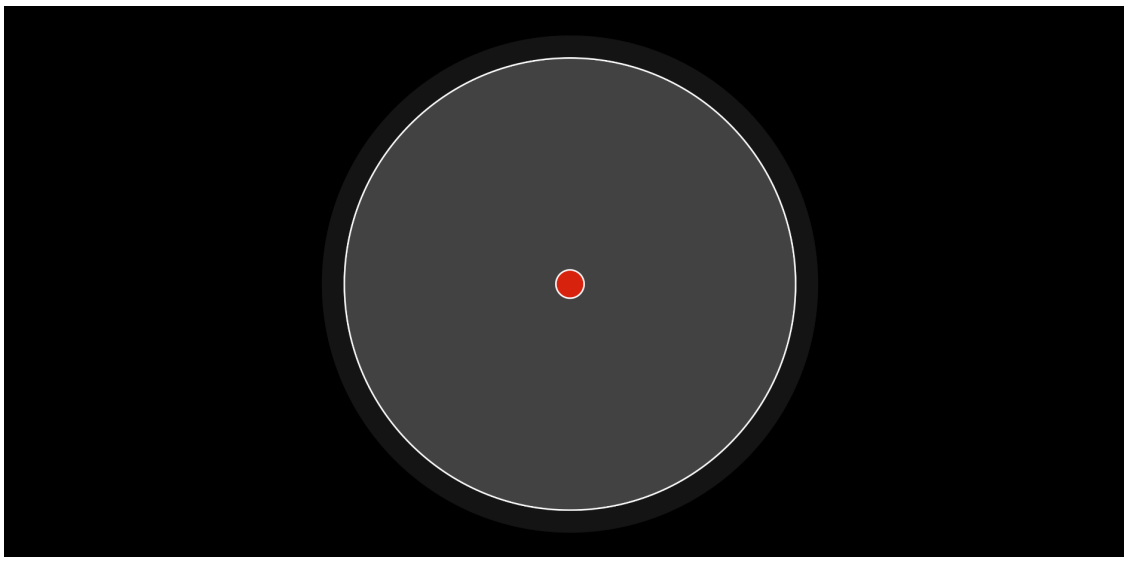

Figure 24: Joystick GUI

As per user tests findings outlined in chapter 5.1, the joystick controller was further developed to display the lidar feed and add a top-down camera feed. The joystick was moved to the right of the screen to make room for the lidar feed. Right

was chosen instead of left because the user's tablet is mounted on the left, which means the right part of the screen is closer to the user’s center field of view. The joystick is displayed on top of a real-time top-down camera feed, so the user can see what the impact of his actions are on the driving behavior of the wheelchair without having to look away from the screen. This has the additional benefit that the wheelchair will drive toward any point you look at on the camera feed.

On the left, the current lidar feed is displayed such that the user can see nearby walls and obstacles, and to know where they are in the current environment, without needing to turn their head. It also provides obstacle detection feedback, where it shows when and where it detected an obstacle. This feedback is important as it tells the user exactly why it stopped moving. If the obstacle detection happened without any feedback, the user would not know which obstacle was detected and how it should be resolved. The rectangle on the right represents the wheelchair and will turn red when any lidar point will be inside of the bounding area if the user were to continue to provide the current joystick input. The joystick pointer circle will also turn red. Once the obstacle is no longer detected, both will turn green.

A big horn button was added in the bottom right, which the user can click to produce a friendly horn sound. This feature can be used to either ask someone for help, or to alert bystanders to move out of its way. The GUI can be seen in figure 25 below.

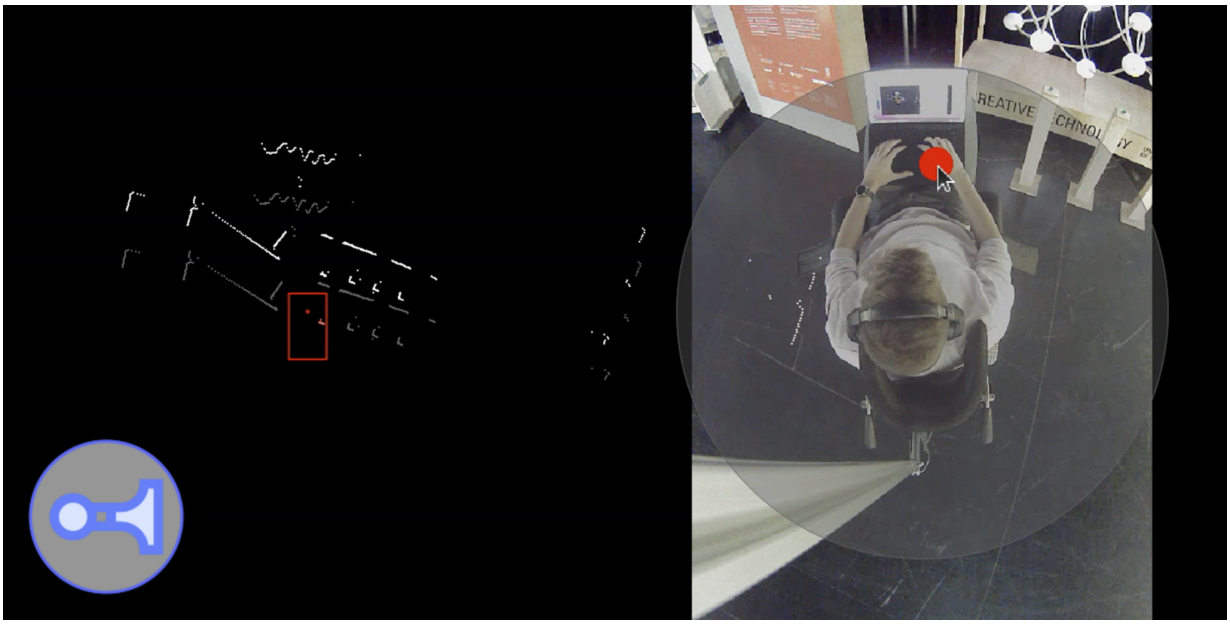

Figure 25: GUI with joystick, top-down camera, lidar feed and horn button

The final GUI is similar to the second, but for this GUI, the left side of the screen is translated and rotated according to the odometry information. This way, the left side resembles a less complex version of a slam map, where the lidar points more

or less stay in place (apart from odometry drift) as the wheelchair moves. The user can see where the wheelchair is in this map and where it is pointing towards.

This also allows the user to click on a target point on the map for the wheelchair to drive to. This can be done by using their tablet to click on a point on the map. Tobii tablets allow users to enable a two-step mouse clicking feature where the user first clicks in the general area where the user wants to click, after which the Tobii will pop up an enlarged version of that area in which the user can click again with more precision. This should allow the user to select their desired target destination with reasonable accuracy, even when the map is relatively small. Alternatively, this two-step click could be implemented into the software directly.

After selecting the destination, the GUI will display the selected point along with a path drawn from the current to the target destination. The user then has to confirm whether they want to execute this path by clicking a second button.

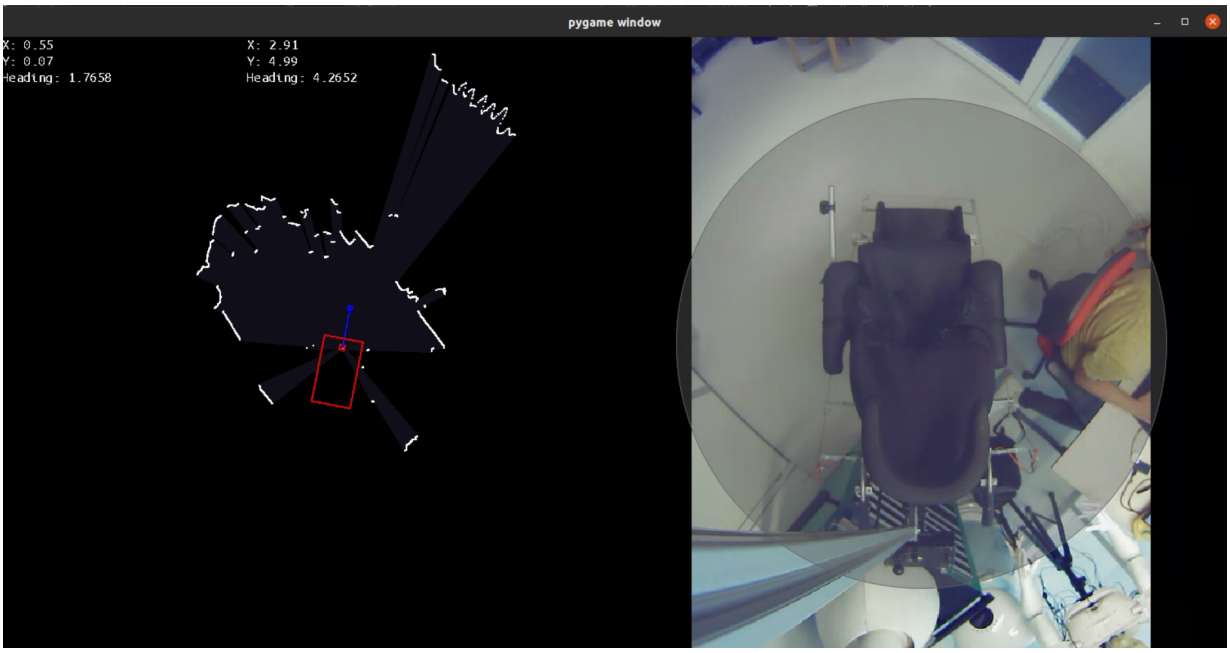

Figure 26: Self-driving GUI

After the action has been confirmed, the wheelchair will start driving towards the target. The joystick area turns into an emergency stop button, which the user can look at for 0.5 seconds to make the wheelchair stop moving. Looking at it for 2 seconds will cancel the current path completely.

The joystick is still included in this GUI to provide a way for the user to move through more tricky areas in which the current self-driving implementation is still struggling.

5. Results and discussion

5.1 User test #1: Virtual joystick

For the first user test, the main aim was to create a simple interface and to get a better understanding of what the user is and is not able to do.

5.1.1 Test setup

Date: 01-12-2022
Location: Designlab, UT
Duration: 20 minutes
Who: Researcher(me), end-user and father
Ethical request filing number: 220197 (see Appendix B)

The first user test took place in the Designlab at the University of Twente, in an open hallway area with an obstacle-free area of about 6 x 10 meters, where the user could freely drive around the wheelchair base without having to mind obstacles.

This test had a few goals. The first goal was to assess the eye control abilities of the user and the Tobii tablet. Controlling a wheelchair with just a joystick requires some level of accuracy and stability in the gaze data, and thus is a good way to test this. Observations before/after the test were also made when the user started up or closed the application, to observe in what ways the user is able to use the tablet and with what accuracy they are able to click on small on-screen elements.

The second objective was to see whether the user would be able to control a wheelchair with a simple eye-controlled joystick. In advance, the expectation was already that this would not be a good solution, but since the user has a lot of experience using eye-tracking devices, there is a chance that the user is able to control the wheelchair in this way, which may be a good alternative control method for areas where self-driving may not work well.

The setup can be seen in figure 27.

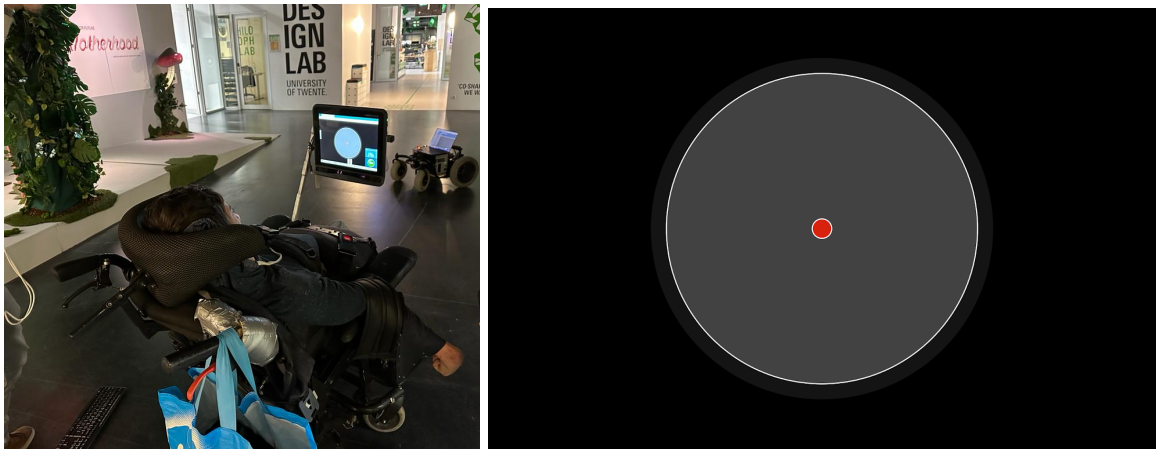

Figure 27: Setup for the first user test.

The user was asked to control the base of the wheelchair from a distance, by using his eyes to control a virtual joystick. The joystick application was built in P5.js and ran in a browser window on the user's Tobii tablet. This input was then sent to the Lattepanda, through an intermediary server, which added around 0.5 seconds of latency. When the user looks into the circle, the x,y coordinates are directly used as inputs for the wheelchair control. When looking outside the circle, the wheelchair stops.

First, the controls were explained to the client, after which the client was instructed to try and drive around the wheelchair to arbitrary locations in the room, and get accustomed to the controls. He was then given around 10-15 minutes to drive around. The researchers noted down observations and helped the user where needed.

5.1.2 Test results

The client struggled to get proper control of the wheelchair. This could be because of the following four reasons:

- The first reason that can be identified is that the user could not see the wheelchair while he was using the joystick. You can only focus your eyes on one thing at any given moment, and thus when controlling the joystick, the user was unable to see what the effect of his actions were on the movement of the wheelchair itself. The moment the user did look at the wheelchair, the mouse moved out of the joystick area and the wheelchair stopped moving.
- The second reason is that the eye-tracking software that is used on the user's tablet does not work well with the joystick. When looking at the same point for around 2 seconds, the tablet registers it as a clicking action. It then first enlarges the area where the click was registered, after which the user can click again with more precision. With the joystick, the user needs to look at the same point for multiple seconds, thus triggering this behavior. This is however a setting in the tablet that the user can switch on/off, after which it is a little bit better, but still not ideal.
- The third reason is that the wheelchair was not equipped with any sensors yet, and thus was unaware of its environment. This led to it bumping into walls a lot, and required the researchers to intervene and correct its course regularly.
- The fourth reason is the amount of effort that is required by the user to control the joystick. Controlling a joystick requires a constant focus for an extensive period of time. During the research this could also be observed, as the user noticeably experienced it as tiring and frustrating.

5.2 User test #2&3: Joystick + Lidar + Camera

Given the findings of the first user tests, a second user test was done which implemented some improvements.

5.2.1 Setup

Date: 27-01-2023 & 03-07-2023
Location: Designlab, UT
Duration: 20 minutes
Who: Researcher(me), end-user and father
Ethical request filing number: 220197 & 230178 (see Appendix B & C)

The second user test took place in the same setting as the first, with a similar procedure. This user test was done twice, once with the user controlling the wheelchair from a distance, and another test a few months later after more safety features were added and the main ethical request was approved, where the user was seated into the wheelchair prototype himself. The same UI was used in both tests, where the only difference was that in the second test, the emergency wire sensor was added, as well as remote stop buttons controlled by the researchers.

The main goal of this test was to see whether the addition of a map and top-down camera would make the joystick controller more usable, and to see whether this type of control would be suitable for situations where the self-driving feature would not work well. For the second test, we also wanted to see how the user responded to physically being in the wheelchair and to relinquish some level of control, and wanted to see whether his physical presence would improve his performance.

The setup is depicted in figure 28 and 29.

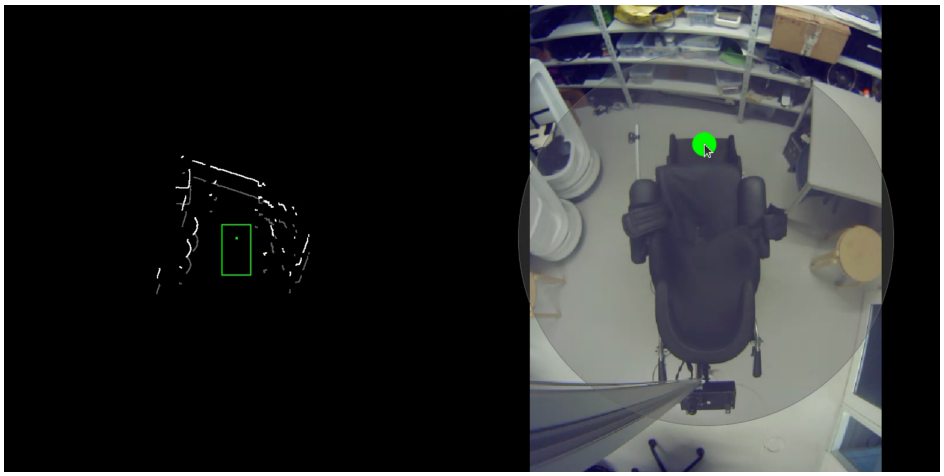

Figure 28: GUI for user test 2

Figure 29: The end user controlling the wheelchair from his tablet

The setup used in the second user test consists of a top-down camera above the wheelchair, which allows the user to get direct visual feedback on what their joystick inputs do to the wheelchair, without needing to look away. Furthermore, the lidar sensor was used to implement basic obstacle detection. The lidar feed (white) is translated and rotated to what its future position and orientation would be in about 1 second, given the current joystick input. If too many lidar points are inside the wheelchair outline, the wheelchair will stop, after which the user can simply move in a non-restricted direction again.

In addition, the communication between the wheelchair and the Tobii tablet was now done directly over the local wifi network, instead of routed via an external server, which greatly reduced the latency.

5.2.2 Results

In the second user test it was found that the performance of the wheelchair increased as compared to the first user test, due to the ability for the user to see in real time what the impact of his actions were. The obstacle avoidance helped with preventing the wheelchair from bumping into things, which made the test more smooth.

In the second test where the user was seated in the prototype, the performance was better, most likely because the user felt more in control. However, the joystick control still did not work very well, partly because of the clicking behavior of the eye tracking software, and the amount of effort it takes to keep focus on the screen, as described in 5.1.2. Thus a different way has to be found where the amount of user input is minimized.

5.3 Self driving interface

While the self-driving feature works, it did not work sufficiently reliable enough to perform a final user test with the current setup. The wheelchair is able to move in a straight line from its current position to the target position on the map, but due to drift in the odometry data, the wheelchair often misses the target destination by a few meters, especially when it has to do a lot of heading correction and cover a relatively long distance. While the encoders work reliably, the drift is mainly caused by the slipping of the wheels. The wheelchair base is quite old, and the tires are worn out and have a large contact area, causing a low friction. The areas in which the wheelchair was driven also had smooth floors, which did not help with this matter. While the IMU is able to correct this to some degree, the drift is still noticeable. A proper implementation of SLAM would solve this issue.

The slipping of the wheels mainly causes an issue when someone is inside the wheelchair, as with this additional weight, the wheelchair is struggling to accelerate, brake, make tight turns or to rotate around its axis. The reason for this is that the weight of the seat and the user is, due to the wheelchair structure, mainly exerted on the front wheels of the wheelchair base. This causes the back wheels to require more friction to be able to push the wheelchair forward. This required friction is not available on the worn out wheels, causing them to slip. This caused the implemented self-driving feature to work decently well with an empty wheelchair, but considerably worse with an occupied wheelchair.

It was hard to tweak the velocity commands in such a way that the wheels would not slip, while applying enough force to move the wheelchair in the desired direction. Possibly, a feedback system with a P(I)D controller could work for this, but since this issue can be solved more easily by simply attaching different wheels, or using a more modern wheelchair base entirely, putting more time and effort into implementing this for this wheelchair seemed not very productive. Also because for further iterations, more focus should be on making a ubiquitous kit that can fit on 90% of (modern) wheelchairs, of which can be assumed that this slipping problem is not present.

5.4.2 Project requirements review

In chapter 2.X, specific requirements for this project were established. These requirements will be reviewed below

MUST HAVE

The end user should be able to move from A to B completely independent of caregivers in a spacious, pre-known environment.

While the self-driving function of the wheelchair works, it only works over short distances and without there being any obstacles in its way. It can move in straight line sequences from different x, y coordinates in the room. The wheelchair will stop for obstacles but not move around them on its own.

The wheelchair should be safe to use and should not expose the user to any unnecessary risks. This can be achieved with the use of simple obstacle detection, as well as accurately being able to detect user input and possibly adding emergency stops. Together with the ethics committee, the risks associated with the user tests should be considered.

Various safety measures were implemented, including the use of a robust control software architecture, the addition of emergency stops, and thorough ethics checks. During user tests, a researcher was always nearby to intervene where needed.

A way for the user to make known his desired destination should be created, most likely using an interface on an eye-tracking tablet.

Multiple GUI interfaces were developed and tested, and it was shown that the user was able to move the wheelchair by only using his eyes.

SHOULD HAVE

A way to map the environment to be able to implement more advanced obstacle detection

This was not achieved during this project. While basic versions of SLAM were successfully implemented, actually using the map proved to be more challenging. Instead, the raw lidar feed was used, which allows the wheelchair to navigate over short distances and avoid obstacles.

A low-effort GUI which allows the user to make known his desired destination without needing to focus on the screen 100% of the time while driving.

Multiple GUIs were designed. The joystick version was not low-effort since it requires full 100% focus and also requires a lot of precision from the user. The map click control, however, is easy to use and only requires a few interactions.

The ability to navigate and drive around in new environments

The self-driving feature uses the current lidar feed and is not dependent on having pre-built maps. However, advanced navigation was not implemented.

COULD HAVE

Long-lasting battery so the user can satisfy his driving needs for a full day (~2 hours of driving)

No extra effort was made to obtain this goal, nor was the battery life measured. During testing, however, the wheelchair was able to keep driving for longer than an hour, so this requirement may have been met.

Robust hard- and software that are not specific for the used wheelchair frame, but is able to be integrated into a variety of existing wheelchairs with low effort.

None of the software/components were specifically tailored to the wheelchair, and thus should be able to be applied to other existing differential-drive wheelchairs. However, while the used odometry solution should be applicable to other wheelchairs, a way to use the built-in odometry sensors of more modern wheelchairs would make the solution even more ubiquitous.

Full path-finding using SLAM, where the user only needs to click on a point on a map, after which the wheelchair fully autonomously drives to the destination.

SLAM and advanced path finding could not be implemented successfully and thus this requirement was not met.

The ability to detect dynamic obstacles (e.g. humans) and be able to notify them to move out of the way in case they are blocking the path.

Since the lidar feed is used, dynamic obstacles are detected and the wheelchair will stop in case someone is blocking its path. The user also has the ability to press a button to make the wheelchair produce a horn sound. This will urge people to get out of the way. More research could be done on how to automate these signals and to implement more or different levels of feedback, e.g. light, voice, ringing, horn, etc.

WON'T HAVE

The ability to drive around outside. Given that there are no walls outside to use as a point of reference, the requirement for sensors to function in different weather conditions, and the safety considerations needed with respect to traffic situations, making the wheelchair be fully operable outside will most likely be out of scope for this project.

While the wheelchair was not tested outside, the chosen hardware should function normally outdoors (although eye-tracking may not work reliably in direct sunlight).The software (joystick and click-on-map controls) are not dependent on static features in the environment and thus should function the same. The emergency stops should also help to make it safer to drive outside, although more safety implementations such as car or sidewalk detection are needed to make it truly usable outside.

6. Conclusion

In this master thesis, research was executed towards developing a safe, easy-to-use and effective way for a (partly) immobile user to drive around in a wheelchair. Literature research on different input methods, related work and control, safety and liability aspects was carried out, after which a wheelchair prototype was developed and tested. While some progress was made towards a fully functional and autonomous wheelchair, the goal of a fully self-driving wheelchair was not achieved. Yet this report could be an important basis for future research.

The research questions defined in Chapter 1 are answered below:

"How can we develop and evaluate an eye-controllable, self-driving wheelchair for (partly) immobile people?"

The development and evaluation of an eye-controllable, self-driving wheelchair involves several key steps. First, a suitable user interface has to be developed which works around the limitations of an eye-tracking device. Secondly, a reliable and robust SLAM system must be implemented to be able to create a map of the wheelchairs environment, in order to give the wheelchair the ability to compute and follow paths. In order to ensure safety, the software should be designed using robust control architectures, and emergency stops should be implemented.

"Which method of input controls suits both the user and application the best when normally functioning muscle control is absent? (eye-tracking, brainwaves, etc.)"

For the case of this user, eye-tracking was most reliable, as normal motor functions in other parts of the body were absent, and the user already has a lot of daily experience with eye tracking. However, for other users, other input methods could yield more accurate results, and especially for emergency stops, having a secondary method that can react faster and be detected with higher accuracy than eye movement would be beneficial.

"What are the best methods of making the wheelchair environment-aware and what software algorithms are needed? (Lidar, 3D camera, etc.)"

SLAM algorithms can be used to create a map of the environment. The minimum choice of sensors would be a Lidar sensor with odometry, but ideally other sensors like a 3D camera would be used in addition, to create a more accurate map. This can help with being able to more accurately detect obstacles and thus making the wheelchair safer.

“How can an interface be developed that will both be efficiently usable by the user, as well as provide enough input for the driving algorithms?”

The interface should be designed in such a way that the user is able to select his target destination with minimal effort. During user tests it was found that operating the wheelchair with a virtual joystick, which requires 100% of the user's focus while driving, does not work very well. Extra confirmation steps may be needed due to inaccuracies in the eye tracking.

7. Future work

In future research, the main focus should be on continuing the effort of making the wheelchair self-driving, by properly implementing slam and path finding, something I was unable to fully get working in this project. In order to achieve this, besides better software, the wheelchair should be equipped with more sensors that can create a more accurate map of the environment. In the current setup, the wheelchair is able to accurately detect and avoid walls, but it struggles with obstacles that lie above or below the lidar range, e.g. tables and chairs. Future research could be done on which exact sensor configuration is needed, while retaining the goal of modularity and affordability.

Other research could focus on making the wheelchair able to safely navigate and drive outside. Indoor, there are many static reference points like walls and furniture that can be used for accurate localization, while outside, there are many open areas with people and traffic, in which case the wheelchair would mostly have to rely on odometry, IMU and GPS data. Furthermore, the wheelchair would have to be able to distinguish between a sidewalk and a road, and between a pebble path and grass. It also needs to be able to be robust enough to not cause any car collisions and not drive into a lake. Eye-tracking also does not work reliably outside on sunny days, for which either a more robust eye-tracking system needs to be developed, or some shade flaps have to be added. Making it work outside was out of scope for this project, but can be an interesting topic for future research.

Lastly, development of the prototype was made harder and more time consuming by the fact that a relatively old wheelchair base was used. The tires were worn out and slipped easily, the solenoid brakes often malfunctioned and turned on randomly, and the lack of built-in odometry sensors made it harder to get reliable odometry information. Therefore it is recommended for future research to use a more modern base.

8. Bibliography

[1] E. Smith and M. Delargy, “Locked-in syndrome,” *BMJ*, vol. 330, no. 7488, Feb. 2005, doi: 10.1136/bmj.330.7488.0-f.

[2] J. León-Carrión, P. van Eeckhout, M. del R. Domínguez-Morales, and F. J. Pérez-Santamaría, “Survey: The locked-in syndrome: a syndrome looking for a therapy,” *Brain Injury*, vol. 16, no. 7, pp. 571–582, Jan. 2002, doi: 10.1080/02699050110119781.

[3] “Locked In Syndrome,” *NORD (National Organization for Rare Disorders)*, Feb. 11, 2015. https://rarediseases.org/rare-diseases/locked-in-syndrome/ (accessed Sep. 27, 2022).

[4] D. J. Kopsky, Y. Winninghoff, A. C. M. Winninghoff, and J. M. Stolwijk-Swüste, “A novel spelling system for locked-in syndrome patients using only eye contact,” *Disability and Rehabilitation*, vol. 36, no. 20, pp. 1723–1727, Dec. 2013, doi: 10.3109/09638288.2013.866700.

[5] Rob@NIUK, “m(eye)DAQ: Enabling Paralysis Sufferers to Communicate,” *NI Community*, Mar. 16, 2015. https://forums.ni.com/t5/Example-Code/m-eye-DAQ-Enabling-Paralysis-Sufferers-to-Communicate/ta-p/3504087 (accessed Jun. 15, 2023).

[6] T. D. Global, “Devices,” *Tobii Dynavox Global*. https://www.tobiidynavox.com/collections/devices (accessed Jun. 15, 2023).

[7] H.-Y. Ryu, J.-S. Kwon, J.-H. Lim, A.-H. Kim, S.-J. Baek, and J.-W. Kim, “Development of an Autonomous Driving Smart Wheelchair for the Physically Weak,” *Applied Sciences*, vol. 12, no. 1, p. 377, Dec. 2021, doi: 10.3390/app12010377.

[8] C. Jayaraman, C. L. Beck, and J. J. Sosnoff, “Shoulder pain and jerk during recovery phase of manual wheelchair propulsion,” *Journal of Biomechanics*, vol. 48, no. 14, pp. 3937–3944, Nov. 2015, doi: 10.1016/j.jbiomech.2015.09.018.

[9] R. K. Megalingam *et al.*, “Self-E: a self-driving wheelchair for elders and physically challenged,” *International Journal of Intelligent Robotics and Applications*, vol. 5, no. 4, pp. 477–493, Oct. 2021, doi: 10.1007/s41315-021-00209-9.

[10] S. P. Parikh, V. Grassi, V. Kumar, and J. Okamoto, “Integrating Human Inputs with Autonomous Behaviors on an Intelligent Wheelchair Platform,” *IEEE Intelligent Systems*, vol. 22, no. 2, pp. 33–41, Mar. 2007, doi: 10.1109/mis.2007.36.
[11] R. H. Abiyev, N. Akkaya, E. Aytac, I. Günsel, and A. Çağman, “Brain-Computer Interface for Control of Wheelchair Using Fuzzy Neural Networks,” *BioMed Research International*, vol. 2016, pp. 1–9, 2016, doi: 10.1155/2016/9359868.
[12] “Home,” *Eyedrivomatic*. https://eyedrivomatic.org/ (accessed Jun. 15, 2023).
[13] M. Dahmani *et al.*, “An Intelligent and Low-Cost Eye-Tracking System for Motorized Wheelchair Control,” *Sensors*, vol. 20, no. 14, p. 3936, Jul. 2020, doi: 10.3390/s20143936.
[14] A. Plotkin *et al.*, “Sniffing enables communication and environmental control for the severely disabled,” *Proceedings of the National Academy of Sciences*, vol. 107, no. 32, pp. 14413–14418, Jul. 2010, doi: 10.1073/pnas.1006746107.
[15] C. Urdiales, B. Fernandez-Espejo, R. Annicchiaricco, F. Sandoval, and C. Caltagirone, “Biometrically Modulated Collaborative Control for an Assistive Wheelchair,” *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 18, no. 4, pp. 398–408, Aug. 2010, doi: 10.1109/tnsre.2010.2056391.
[16] E. Miranda, “Moscow Rules: A Quantitative Exposé,” in *Lecture Notes in Business Information Processing*, Cham: Springer International Publishing, 2022, pp. 19–34. Accessed: Jun. 15, 2023. [Online]. Available: http://dx.doi.org/10.1007/978-3-031-08169-9_2
[17] PassengerSelfService, “Amsterdam Schiphol trials autonomous wheelchairs,” *PASSENGER SELF SERVICE*, Feb. 26, 2023. https://www.passengerselfservice.com/2023/02/amsterdam-schiphol-trials-autonomous-wheelchairs/ (accessed Jul. 01, 2023).
[18] H.-Y. Ryu, J.-S. Kwon, J.-H. Lim, A.-H. Kim, S.-J. Baek, and J.-W. Kim, “Development of an Autonomous Driving Smart Wheelchair for the Physically Weak,” *Applied Sciences*, vol. 12, no. 1, p. 377, Dec. 2021, doi: 10.3390/app12010377.
[19] R. Alkhatib, A. Swaidan, J. Marzouk, M. Sabbah, S. Berjaoui, and M. O.Diab, “Smart Autonomous Wheelchair,” in *2019 3rd International Conference on*

Bio-engineering for Smart Technologies (BioSMART), Apr. 2019. Accessed: Jul. 02, 2023. [Online]. Available: http://dx.doi.org/10.1109/biosmart.2019.8734264
[20] “Simultaneous Localization and Mapping (SLAM),” in *Encyclopedia of Ocean Engineering*, Singapore: Springer Nature Singapore, 2022, pp. 1713–1713. Accessed: Jul. 02, 2023. [Online]. Available: http://dx.doi.org/10.1007/978-981-10-6946-8_300725
[21] “How does it work and which are the types of eye tracking?,” *BuscaEU*, Nov. 07, 2019. https://www.brainlatam.com/blog/how-does-it-work-and-which-are-the-types-of-eye-tracking-816 (accessed Jul. 03, 2023).
[22] M. J. Mataric, *The Robotics Primer*. MIT Press, 2007.

9. Appendices

Appendix A: User interview questions

This interview is meant to get more clarity in what is actually required.
Why does the client think he needs a self-driving wheelchair?
Where does the client intend to use the self-driving wheelchair?
What is the client able/not able to do?

The interview will be semi-structured.
Since the client has a harder time to answer open-questions, questions directed towards the client will be formulated as yes/no questions as much as possible.
Yes/no questions can possibly be answered by the client using the 'fist' method where the client looks at my fist when he agrees, or looks away from my fist when he disagrees.

Questions will be asked in Dutch

Globale vragen over de cliënt om de cliënt beter te leren kennen

1. Hoe oud is de user?
2. Hoelang heeft de user al Locked-in? Hoe is het ontstaan?
3. Wat vindt de user leuk om te doen? (hobby's)
4. Wat zou de user willen doen maar niet kan?
5. Wat vindt de cliënt het grootste nadeel van het hebben van locked-in?
6. Voelt de cliënt zich gezien / buitengesloten / gerespecteerd / begrepen door de rest van de maatschappij?
7. Kan ik jullie naam en/of fotos gebruiken in mijn verslag
8. Breekt de user snel dingen?

Vragen m.b.t. de vaardigheden van de cliënt

9. Wat is de exacte aandoening? (Is het 'gewoon' locked-in, of een variatie erop?)
10. Over welke lichaamsdelen (afgezien van de ogen) en in welke mate heeft de user controle?
11. In hoeverre heeft de user controle over zijn ogen?
    - Patiënten met locked-in kunnen vaak hun ogen nog bewegen, maar is dit in dezelfde mate als iemand zonder locked-in? Of juist beter omdat ze regelmatig oefenen met een eye-tracking tablet?
    - Wat kan de user buiten het tekenen nog meer met zijn ogen? (typen, gamen)
    - De user heeft ook vormen van spasme. In hoeverre beïnvloed dit zijn oogfuncties? Schieten zijn ogen soms een andere kant op bij een spasme? Kan hij 100% van de tijd de focus behouden?
12. In hoeverre werken de zintuigen? Kan de hij alles zien / horen / voelen / proeven / ruiken?
13. Er is al onderzoek gedaan naar een rolstoel die met neus-snuiven bestuurd kan worden. Zou de cliënt dit ook kunnen?
14. Heeft de cliënt op intellectuele vlakken een beperking?

15. Kan de cliënt Engels (voor het geval engelse software gebruikt moet worden)
16. Is er altijd een begeleider in de buurt (mocht er ondanks alle veiligheidsmaatregelen toch iets mis gaan met de rolstoel)

Vragen m.b.t. de wensen van de cliënt

17. Waarom wil de cliënt een zelfrijdende rolstoel?
    - Zelfstandiger worden?
    - Meer vrijheid willen?
    - Andere mensen willen ontlasten door minder afhankelijk van anderen te zijn?
    - Niet hoeven wachten op hulp maar zelf direct ergens heen kunnen?
    - Meer betrokken zijn bij de maatschappij?
18. Wat voor oplossingen heeft de cliënt al geprobeerd? (Parrot etc.). Wat waren daar de bevindingen (plus/min punten) van?
19. Hoe denkt de cliënt de rolstoel het liefst te besturen?
    - Volledige pathfinding? (kijk naar een punt op een kaart, computer berekent route en gaat volledig automatisch daarheen. Cliënt geeft controle uit handen en doet zelf niet veel)
    - Route tekenen? (zelf een lijn tekenen van een route waar de rolstoel heen moet, rolstoel volgt zoveel mogelijk deze lijn Cliënt behoud nog een beetje controle, maar rolstoel doet het meeste)
    - Joystick? (met ogen een virtuele joystick besturen, altijd zelf de controle houden waar de rolstoel heen rijdt. Cliënt behoud zelf de controle)
    - Of juist niet met de ogen maar met brainwaves of een ander lichaamsdeel dat nog wel werkt?
    - *In alle opties grijpt de rolstoel in als er een drempel/obstakel dichtbij is
20. Wat voor functies denkt de cliënt dat de rolstoel moet hebben?
    - Obstacle detection
    - Auto-path finding
    - Knipperlichten voor omstanders
    - “Fietsbel”/”toeter” om omstanders te waarschuwen
    - Achteruitrij camera
    - Licht (ledstrips, koplampen) voor in het donker
    - Auditory feedback, bijvoorbeeld als er een obstakel dichtbij is

Vragen m.b.t. het beoogde gebruik van de rolstoel

21. Op welke locaties denkt de cliënt de rolstoel te gaan gebruiken?
    - Als het alleen thuis en dagbesteding is, dan zijn er minder complexe hardware/algoritmes nodig dan wanneer de cliënt er ook buiten over een drukke straat met verschillende weersomstandigheden wil rijden.
    - Waar bevindt de cliënt zich het vaakst? Als de cliënt zegt dat hij het op straat wil gebruiken, maar zich daar maar 2 uur per week bevindt, dan is dat misschien overbodig.
    - Hoe zien deze locaties eruit? Zijn het grote of juist smalle ruimtes?
22. Voor hoeveel uur per dag denkt de cliënt te rijden met de rolstoel? (accu grootte)

Praktische vragen

23. Hebben jullie nog hardware liggen dat geleend kan worden? (eye-tracker, rolstoel-stoel)
24. Waar draait de Tobii tablet op (Windows denk ik), Kan ik elke windows applicatie erop runnen of zitten er limitaties aan mbt opslag grootte / cpu performance etc.
25. Kennen jullie andere mensen die baat kunnen hebben bij een soortgelijke rolstoel (Locked in of anders)? En zouden jullie mij met hun in contact kunnen brengen?
26. Zijn er restricties voor het gewicht van de rolstoel? (ivm met vervoer, (trap)liften, moet over drempels te tillen zijn door begeleider, etc.)
27. Hardware specs (Welke poorten zijn beschikbaar op zijn tablet, windows versie, CPU/GPU/RAM, etc.)

Appendix B: Ethics consent form for user test 1

Information Letter

The purpose of this user test is to be able to iteratively develop the prototype, where end-user feedback is used to determine which parts of the prototype work well or which parts need improvements. The project involves the development of a self-driving wheelchair for a patient with locked-in.

The user test will take around 15 minutes. During the test, the participant will be asked to remotely control the wheelchair by interacting with a user interface on a screen.

The research has been reviewed by the Ethics Committee and therefore should not have any unforeseeable risks.

The participant can withdraw from the interview at any time by, in case of the locked-in patient, staring at a pre-agreed on object in the room, or by letting his caretaker step in.

The user test will not be audio- or video-recorded. But interaction data (eye movements, wheelchair location, etc.) may be recorded. Notes will be taken regarding the observed interaction and short follow up questions. The collected data will be used in the development of the project and will be anonymously documented in the final thesis. The participant has the right to request access and rectification or erasure of the personal data.

The obtained data will only be used among the team members working on this project and will be stored on a private Google Drive. If desired by the interviewee, the data can be anonymized in either or both the storage and publication of the data. Data will be deleted after the report has been finished in about 8 months from now.

Contact details:

Researcher: Sjoerd de Jong (MSc I-TECH student), s.dejong@student.utwente.nl

Ethics committee(for complaints): ethicscommittee-cis@utwente.nl

Consent Form for User Tests for the Development of a Self-Driving Wheelchair

YOU WILL BE GIVEN A COPY OF THIS INFORMED CONSENT FORM

Please tick the appropriate boxes	**Yes**	**No**
Taking part in the study		
I have read and understood the study information dated [20/12/2022], or it has been read to me. I have been able to ask questions about the study and my questions have been answered to my satisfaction.	○	○
I consent voluntarily to be a participant in this study and understand that I can refuse to answer questions and I can withdraw from the study at any time, without having to give a reason.	○	○
I understand that taking part in the study involves the recording of unidentifiable data	○	○
I understand that information I provide will be used for the final project report	○	○
I understand that personal information collected about me that can identify me, such as [e.g. my name or where I live], will not be shared beyond the study team.	○	○
Future use and reuse of the information by others		
I give permission for the interaction data that I provide to be archived in a private Google Drive so it can be used for future research and learning.	○	○
	○	○
	○	○

Signatures

_____	_____	_____
Name of participant [printed]		
	Signature	Date

For participants unable to sign their name, mark the box instead of sign

UNIVERSITY OF TWENTE.

Appendix C: Ethics consent form for user test 2

Information Letter

The purpose of this user test is to test the prototype, where end-user feedback is used to determine which parts of the prototype work well or which parts need improvements. The project involves the development of a self-driving wheelchair for a fully immobile patient.

The user test will take around 15-60 minutes. During the test, the participant will take place in the wheelchair, and asked to move himself around by interacting with a user interface on a screen. Various safety measures have been implemented to reduce potential risks to the patient in case of a soft-or-hardware malfunction, including a safety bumper and multiple dead-man switches.

The research has been reviewed by the Ethics Committee and therefore should not have any unforeseeable risks.

The participant can withdraw from the interview at any time by making this known or by having his caretaker step in.

The user test will not be audio- or video-recorded. But interaction data (eye movements, wheelchair location, etc.) may be recorded. Notes will be taken regarding the observed interaction and short follow up questions. The collected data will be used in the development of the project and will be anonymously documented in the final thesis. The participant has the right to request access and rectification or erasure of the personal data.

The obtained data will only be used among the team members working on this project and will be stored on a private Google Drive. If desired by the participant, the data can be anonymized in either or both the storage and publication of the data. Data will be deleted after the report has been finished in about 4 months from now.

Contact details:

Researcher: Sjoerd de Jong (MSc I-TECH student), s.dejong@student.utwente.nl

Ethics committee(for complaints): ethicscommittee-cis@utwente.nl

Consent Form for User Tests for the Development of a Self-Driving Wheelchair

YOU WILL BE GIVEN A COPY OF THIS INFORMED CONSENT FORM

Please tick the appropriate boxes	**Yes**	**No**
Taking part in the study		
I have read and understood the study information dated [16/03/2023], or it has been read to me. I have been able to ask questions about the study and my questions have been answered to my satisfaction.	○	○
I consent voluntarily to be a participant in this study and understand that I can refuse to answer questions and I can withdraw from the study at any time, without having to give a reason.	○	○
I understand that taking part in the study involves the recording of unidentifiable data	○	○
I understand that information I provide will be used for the final project report	○	○
I understand that personal information collected about me that can identify me, such as [e.g. my name or where I live], will not be shared beyond the study team.	○	○
Future use and reuse of the information by others		
I give permission for the interaction data that I provide to be archived in a private Google Drive so it can be used for future research and learning.	○	○

Signatures

_____	_____	_____
Name of participant [printed]	Signature	Date

For participants unable to sign their name, mark the box instead of sign

UNIVERSITY OF TWENTE.