Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Department of Computer Science

Chair of Software Engineering

MASTER'S THESIS

February 28th, 2015

Robot control by user tracking with a laser range scanner

Author:
Ivo Steinmann

ID: 02-906-519
EMail: ivost@ethz.ch

Supervisor:
Dr. Jiwon Shin
Prof. Dr. Bertrand Meyer

Zusammenfassung

Die Verwendung eines Rollators durch gehbehinderte Personen ist mittlerweile stark verbreitet. Diese Arbeit befasst sich mit einer Hightech-Erweiterung für Rollatoren, welche den Alltag für solche Menschen erleichtern soll.

Die Erweiterung besteht aus verschiedenen Sensoren wie Laser Scanner, Neigungsmesser und Geschwindigkeitsmesser, und zwei Motoren für den Antrieb. Der Laser Scanner liefert die Daten, um die Beinbewegung der Person und den Abstand zum Rollator zu messen. Mit diesen Informationen und den Daten von den anderen Sensoren berechnet der Kontroller die optimale Leistung für die Motoren, damit das Vorwärtskommen trotz Rollator möglichst ohne zusätzlichen Kraftaufwand möglich ist. Dabei werden auch Steigungen und Gefälle berücksichtigt.

Der erste Prototyp wird in drei Altersheimen zusammen mit den Bewohner getestet. Die Erkenntnisse werden anschliessend ausgewertet und die nötigen Schlussfolgerungen gezogen. Mit den Erfahrungswerten aus diesen Tests werden mögliche Verbesserungen und Korrekturen präsentiert.

Abstract

The use of a walker by people with reduced mobility is common nowadays. This thesis deals with a high-tech extension for walkers, which should make life easier for those people.

The extension consists of various sensors such as a laser scanner, inclinometer and speedometer and two engines for the electric propulsion. The laser scanner provides the data to measure the leg movement of the person and the distance to the walker. With this information and the data from the other sensors, the controller calculates the optimum performance for the motors, so walking is possible without additional effort despite of the rollator. Also climbs and descents are taken into account.

The first prototype is tested in three nursing homes along with the residents. The findings are then evaluated. With the experience gained from these tests improvements and corrections are presented.

Acknowledgements

I would like to express my special thanks of gratitude to my supervisor Dr. Jiwon Shin for the helpful support and the interesting discussions as well as our Professor Prof. Dr. Bertrand Meyer who provided me the possibility to work on this robotic project of his group.

iHomeLab and especially Marcel Mathis I would like to thank for the efficient and helpful cooperation.

This project was partially funded by the Hasler Foundation under the SmartWorld program and received a donation of inclinometer sensor from the Pewatron Company.

Contents

List of Figures

List of Tables

List of Listings

1 Introduction

RoboScoop is a research project of the Chair of Software Engineering at ETH Zurich and iHomeLab at Hochschule Luzern; the Autonomous Systems Lab at ETH Zurich serves as a project advisor, bringing its own experience in autonomous robots. The aim of RoboScoop is to improve the tools and techniques for developing robotics software. The main demonstrator application is the SmartWalker robot for Ambient Assisted Living [38].

1.1 Motivation

The society is ageing and at the same time there is a growing lack of caregivers for older persons. Handicapped people are in a difficult position, because mobility is an important asset to maintain a certain degree of independence and to keep contacts with the social environment. For these people, the walker is an important everyday tool. Since their introduction, these devices have made hardly any technical progress, especially when you consider today's electronic and information technology.

SmartWalker is a high-tech extension of a walker that aids people of reduced mobility moving around, so that they can cope with everyday life better. The walker is equipped with sensors and actuators and is designed to function both, autonomously and non-autonomously. Itten [18] proposed a solution for the autonomous mode. The non-autonomous assist mode, that is developed in this project, promises to help people moving outdoors and indoors.

This project addresses the challenges in the non-autonomous assist mode, in particular, controlling the walker's velocity according to its user's movement. The assist mode is transparent to the user and requires no active user control. It reduces the resistance when pushing the walker, especially when moving uphill or downhill. For example, it is possible to carry purchases with less effort.

The controller of the assist mode is using data from leg tracking, speed, brake lever and inclinometer sensors. The latter two sensors are evaluated and implemented into the existing hardware interfaces in terms of this thesis.

1.2 Tasks

This work involves processing laser data to detect and track the user's leg movement, the assembly of an additional tilt sensor (inclinometer) and brakes, and writing a control loop for the walker to move accordingly. It is required to develop and implement various algorithms in the fields of leg detection and tracking and controlling.

The prototype is evaluated evaluated in elderly homes together with real people.

1.3 Related Work

In this thesis we introduce a inclinometer sensor and use this data in our controller to adjust the support power. Tausel *et al.* [40] presents a model for smart walker interaction on slopes. The parameters for this interaction model are obtained from a laser range finder for tracking the legs and an inertial measurement unit, which can measure the pitch and the roll angles. They integrate the model into a closed control loop that operates the engines. Overall this work can be compared with our solution, but they use more expensive sensors to reach the same.

Arlindo *et al.* [12] propose a robotic walker for clinical applications in controlled rehabilitation, but also for domiciliary use. Their walker has a three-dimensional force sensor at the handlebars

and a laser scanner between the wheels to provide user-walker interaction data used for generating navigation commands. The authors record this data for clinical analysis and for fine-tuning of different walking training and rehabilitation programs. This rollator is only suitable for flat surfaces.

The smart walker of Postolache *et al.* [32] is designed to assist in physiotherapy sessions for gait analysis and recovery. They use handlebars with piezo-resistive force sensors for measuring the force applied by the user to the walker. It is an option to evaluate similar sensor for our walker.

Geunho *et al.* [21] developed a walker that is controlled exclusively by leg movement. The authors realized a tracking scheme to estimate and predict the locations of the user's legs and body. With this informations they control the motions of the robot in real time. The big difference to our walker is not only the control, but also the structure of the frame. The user walks in the centre of the walker and the three engines are located in a triangle around the user.

Kim, Chung, and Yoo [20] propose a method for a mobile robot to detect and follow human legs. This method is explained in short in Section 3, as we use ideas from their leg detection method for our own implementation. Besides that, a service robot function would be an ideal extension for our walker.

1.4 Structure of Thesis

The thesis is structured into seven parts. In Section 1.3 related work concerning leg detection and tracking is presented. This is followed by an introduction (Section 2) to the hardware and software, developed and used for the SmartWalker project. In Section 3), we take a closer look at the LiDAR and the used leg detection algorithms. Next, there is a short attempt to use the LiDAR also for environment tracking (Section 4). Section 5 is devoted to the two different control modes, the PID and the speed to power (STP) mode. Finally, in Section 6.3, the experiments and the analysis are presented of what we have done in retirement homes. This is followed by a final conclusion (Section 7).

2 SmartWalker

This section explains the setup of the SmartWalker and its components. Some of the sensors were already present when this project was started, like engines, motor controller and LiDAR. Brake sensors and inclinometer were introduced as consequence during this work.

In the first part we present the used hardware components, and in the second part the different software modules.

2.1 Hardware

(a) Unmodified walker

(b) Walker after the modifications

Figure 1: Trionic Veloped Walker

SmartWalker is a modification and extension of an existing walker. The underlying walker (Figure 1) is produced by the swedish company Trionic[1]. The original usage area is located in the outdoor field. The big wheels with rough tires allows one to use the walker also in rough terrain. In the context of this thesis, the main advantage of this massive frame for us was the possibility to mount industrial motors and sensors without miniaturisation and specialisation.

Roboscoop is a collaborative research project between iHomeLab, *the Think Tank Research Centre for Building Intelligence of the Lucerne University of Applied Sciences and Arts* and the Chair of Software Engineering of ETH Zurich. In terms of this research project, the Trionic walker was extended with some basic set of sensors and actuators as two hub engines, a LiDAR sensor and a rotatable camera. These components allow developers to build different applications to make the walker behave smart.

Figure 2 shows a complete overview of the used hardware components. The core is the motherboard (Section 2.1.1). It is connected to two motor controller (Section 2.1.2), two brakes (Section 2.1.3), a laser scanner (Section 2.1.4), an inclinometer sensor (Section 2.1.5), a 3D camera mounted on a servo motor (Section 2.1.6) and a tablet computer (Section 2.1.7).

[1] *Trionic Veloped - the walker for active people.* Trionic. URL: http://www.trionic.se/. Veloped is the modern alternative to a rollator - offering greater access and better comfort.

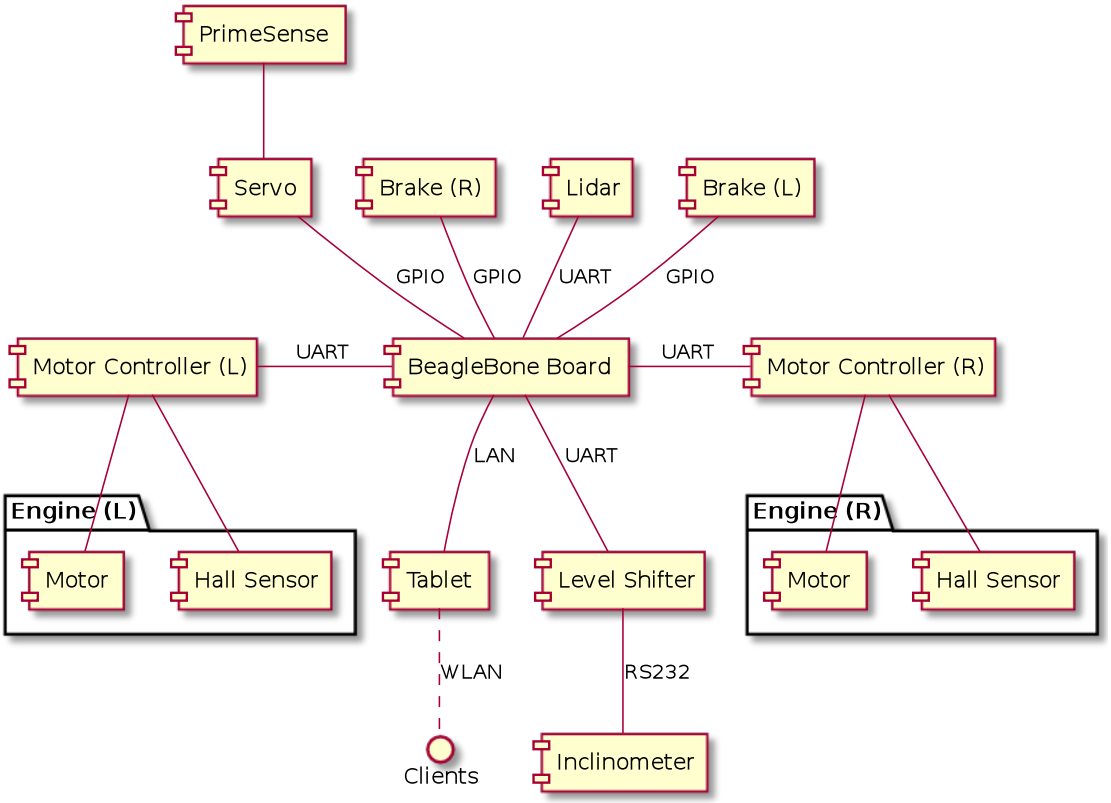
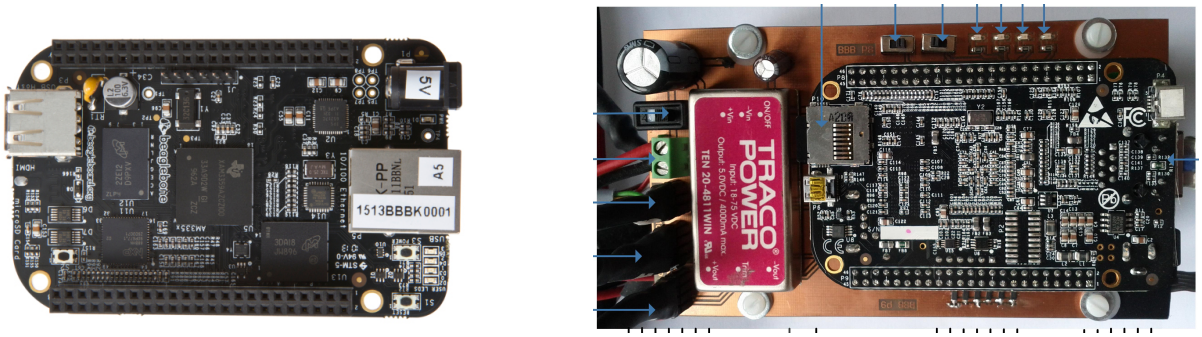


Figure 2: Hardware component overview

2.1.1 Controller Motherboard

The controller is the core component of the SmartWalker. It is a BeagleBone Black[2] (Figure 3a) computer mounted on a motherboard (Figure 3b). This motherboard provides a 12V power connector, multiple UART and GPIO interfaces to connect the actuators and sensors, and a $5V$ USB power outlet. It is an in-house development from iHomeLab.



(a) BeagleBone Black

(b) Motherboard with BeagleBone developed by iHomeLab

Figure 3: Motherboard for BeagleBone Black

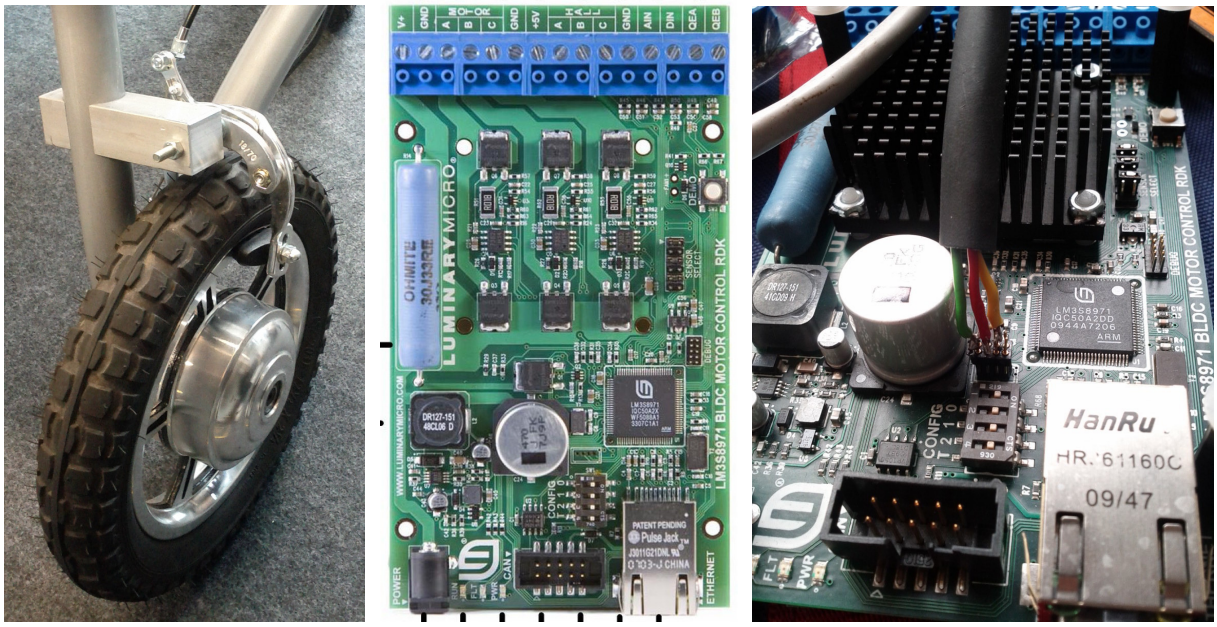
[2] *BeagleBone Black*. BeagleBoard.org. URL: http://beagleboard.org/BLACK.

2.1.2 Engines & Motor Controller & Hall Effect Sensors

The most important actuators are the two hub engines located in the two back side wheels (Figure 4a). Besides the motor, there is also a hall effect sensor[3] contained inside the hub. This sensor produces one tic per degree when the wheel is rotating. With this information it is possible to compute the rotation speed of the wheel and therefore the speed of the robot (see Section 2.2.4).

Both engines are controlled by a Stellaris LM3S8971 BLDC Motor Control Board[4] from Texas Instruments (Figure 4b and 4c). The Motor Control Board is connected with the battery directly and with the BeagleBone over UART. It supports two modes of operation, PowerMetric and SpeedMetric.

- **PowerMetric**: The controller keeps the power to the motors constant. Therefore, the speed of the walker is slower if there is higher resistance, or faster if there is less resistance. The speed mode is used by the speed to power (STP) controller module (Section 5.3).
- **SpeedMetric**: The controller holds the speed of the motors constant. More power is consumed at higher resistance, less otherwise. The power mode is used by the PID controller module (Section 5.2).



(a) Hub Engines (b) Motor controller board (c) Motor controller board (installed)

Figure 4: Engine controller board and hub engine

[3] *Hall effect sensor*. Wikipedia. URL: http://en.wikipedia.org/wiki/Hall_effect_sensor. A Hall effect sensor is a transducer that varies its output voltage in response to a magnetic field. Hall effect sensors are used for proximity switching, positioning, speed detection, and current sensing applications.

[4] *Brushless DC (BLDC) Motor RDK*. Texas Instruments. URL: http://www.ti.com/tool/RDK-BLDC.

2.1.3 Brakes

In the pilot study, which was done as part of the thesis written by Itten [18], people complained about missing brakes. Therefore, we mounted two Tektro EL550-RS[5] brake levers known from electrical bicycles (Figure 5a) and the engineers from iHomeLab have designed mounts for the calliper brake (Figure 5b) to the Trionic frame.

The two brake levers are mechanically connected to two side pull dual pivot brakes[6], one for each motorized wheel. In addition, both levers have a reed[7] sensor included. These two sensors are connected to the motherboard over GPIO. It allows developers to read out the state of the brake. But since reed sensors are binary only, the possible states are limited to `pulled` and `released`.

(a) Tektro EL550-RS brake lever

(b) Dual pivot calliper brake

Figure 5: Calliper brake and levers

[5] *Tektro EL550-RS Brake Lever*. Tektro. URL: http://www.tektro.com/_english/01_products/01_prodetail.php?pid=199&sortname=Lever&sort=1&fid=3.

[6] *Rim brake*. Wikipedia. URL: http://en.wikipedia.org/wiki/Bicycle_brake#Rim_brakes. Rim brakes apply the force by friction pads to the rim of the rotating wheel.

[7] *Reed switch*. Wikipedia. URL: http://en.wikipedia.org/wiki/Reed_switch. The reed switch is an electrical switch operated by an applied magnetic field.

2.1.4 LiDAR

The Light detection and ranging[8] (LiDAR) measures the distances to objects in discrete intervals within a view range of 360°. It is one of the important sensors for this project because it is used for leg detection in Section 3 and for environment tracking in Section 4.

In general, these sensors are expensive. The walker uses the scanner from the affordable Neato Robotics[9] XV-11 robotic vacuum cleaner. There are various internet websites (so called hacker spaces), which explain how to use the LiDAR from the XV-11 robot in own robotic projects[10].

The XV-11 LiDAR is connected as UART device to the motherboard. The mounting position is in the middle of the SmartWalker, respectively in the center of the triangle shaped by the three wheels, around $20cm$ above the floor (Figure 6). The XV-11 have got a resolution of 1° and a full 360° scan takes 0.25 seconds, therefore the scan frequency is $4Hz$. The range is between $2cm$ and $4m$.

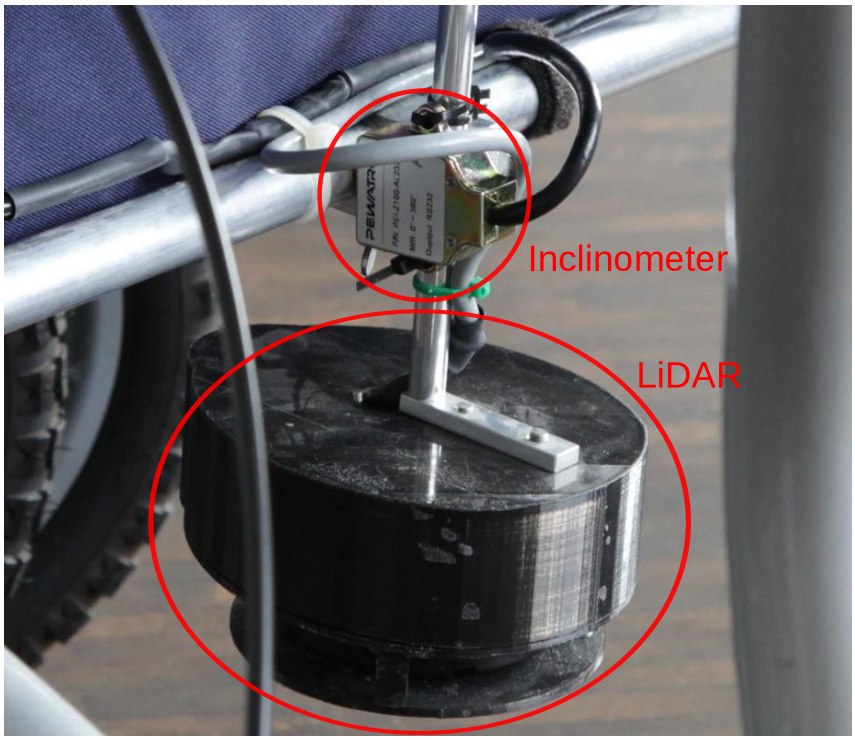


Figure 6: LiDAR of the SmartWalker

[8] *Lidar*. Wikipedia. URL: http://en.wikipedia.org/wiki/Lidar. Lidar (also written LIDAR, LiDAR or LADAR) is a remote sensing technology that measures distance by illuminating a target with a laser and analyzing the reflected light.

[9] *Neato Robotics*. Neato Robotics. URL: http://www.neatorobotics.com.

[10] *Hacking the Neato XV-11*. Wikispaces. URL: https://xv11hacking.wikispaces.com/.

2.1.5 Tilt Sensor/Inclinometer

The ground is rarely flat. For persons with reduced mobility gradients are exhausting, especially if one have to push a walker uphill. For that reason, the motors must be controlled depending on the slope. Therefore, as part of this thesis, we introduced an inclinometer to measure the pitch of the ground. The criteria for a sensor are:

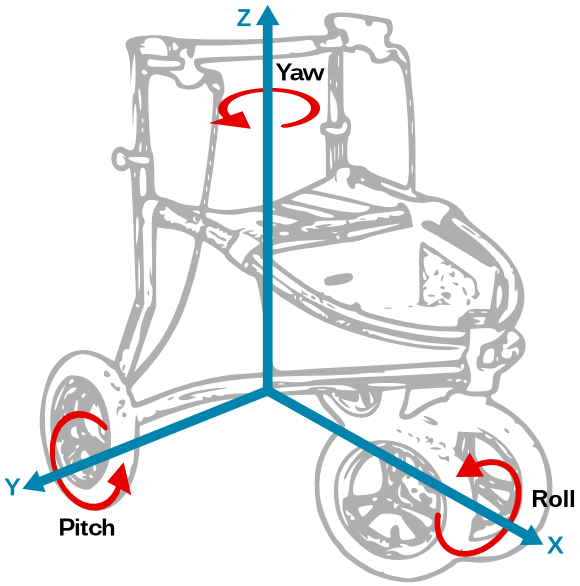
- Low power requirements
- Voltage of $5V$ or $12V$ (supported power outlets of the motherboard)
- Resolution below $0.2°$ (smooth changes of the ground)
- Serial protocol support (UART or RS-232)
- Low price

We have selected the Pewatron PEI-Z100-AL-232-1 360° inclinometer[11] (Figure 7a), because it meets the defined requirements. The resolution of this sensor is $0.1°$ and it is powered with $5V$. It is connected to an UART port of the motherboard over an UART-to-RS-232 level shifter. The level shifter is required, because BeagleBone Black has only UART ports, but no RS-232 connectors. While for both standards, the transmission protocol is the same, UART works with a voltage level between $-3.3V$ and $0V$ and RS-232 between $-12V$ and $0V$.

Other options include the PEI-Z245-AL-232-1-17, 2 axis $\pm$ 45°[12] sensor, because it would allow us to measure the roll of the ground. If the ground has a longitudinal gradient, the walker drifts either to the left or the right side. With the knowledge about the roll angle it would be possible to compensate the drift. Figure 7b shows a schematic view of the rollator with the names of the different rotation axis.

The inclinometer is mounted on the top of the LiDAR (Figure 6) and it is used by the controller explained in Section 5.

(a) Pewatron PEI-Z100-AL-232-1 360° inclinometer



(b) Pitch and roll

Figure 7: Inclinometer

[11] *PEI-Z100-AL-232-1 360° inclinometer*. PEWATRON Online Shop. URL: http://shop.pewatron.com/search/pei-z100-al-232-1-360%C2%B0-inclinometer.htm.

[12] *PEI-Z245-AL-232-1-17, 2 axis ± 45°*. PEWATRON Online Shop. URL: http://shop.pewatron.com/search/pei-z245-al-232-1-17--2-axis.htm.

2.1.6 3D Camera

For 3D sensing, there is a PrimeSense Carmine[13] (Figure 8a) sensor. Figure 8b shows the sensor mounted on a small servo motor at the height of the handlebar . The PrimeSense 3D sensing technology gives developers the ability to observe a scene in three dimensions. The field of view is 57.5°x45°, the VGA depth map has a resolution of 640x480 pixels and there is audio and color support. The sensing range is between $0.8m$ and $3.5m$. With the servo motor it is possible to rotate the camera by 360°.

The PrimeSense Sensor is connected via USB to the Tablet or to the BeagleBone Black. The servo motor itself is connected via GPIO to the motherboard.

(a) PrimeSense Carmine sensor

(b) PrimeSense mounted on a servo motor

Figure 8: PrimeSense sensor

2.1.7 Tablet Computer

For processing computationally intensive algorithms such as 3D image analysis or as display for some user interface, there is a tablet computer available. It is mounted on the top of the handlebars. The tablet computer is connected via LAN to the BeagleBone and has a direct connection to the main battery. The connection to the battery allows the tablet to be permanently switched on. With the built in Wifi device, the tablet can also act as wireless Access Point and bridge[14] to the internal network. Figure 9 shows the mounted tablet.

Figure 9: Tablet computer mounted on the SmartWalker

[13] *PrimeSense*. Wikipedia. URL: http://en.wikipedia.org/wiki/PrimeSense.

[14] *Bridging (networking)*. Wikipedia. URL: http://en.wikipedia.org/wiki/Bridging_(networking). Network bridging is the action taken by network equipment to create an aggregate network from either two or more communication networks.

2.2 Software

The controller software is structured into multiple thematic packages. These packages are `Configuration` (Section 2.2.1), `Terminal` (Section 2.2.2), `ROS` (Section 2.2.3), `Position` (Section 2.2.4), `Threading` (Section 2.2.5), `Drivers` (Section 2.2.6), `LegDetection` (Section 3) and `Controller` (Section 5).

Figure 10 shows an overview of all packages and the owned classes. This UML diagram focuses on the packages and the contained classes, therefore the definition of methods, fields and relations is omitted. Besides these that, there are also firmware modules running on the engine controllers or the tilt sensor, but these are also omitted in the following introduction.

The operating system used on BeagleBone is Linux Ubuntu Trusty 14.04 for ARMhf[15]. The controller software is written in C++ and structured and compiled as ROS node.

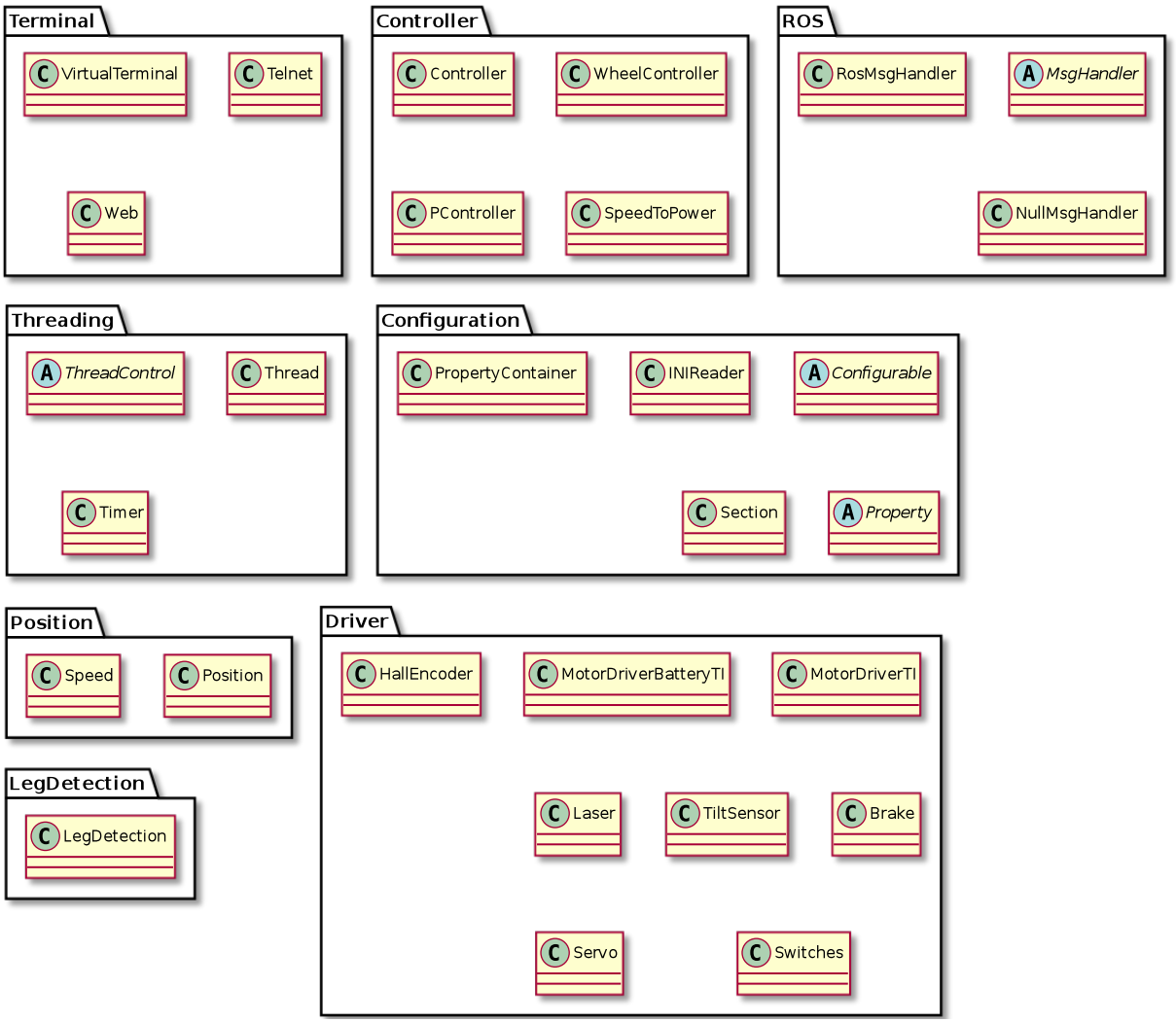


Figure 10: UML diagram of all packages and important classes

[15] *Prepackaged ARM HF Linux Images*. ARMhf. URL: http://www.armhf.com. Easy to deploy ARM HF Linux images for BeagleBone, ODROID-XU, and Wandboard devices.

2.2.1 Configuration

The configuration module provides an interface to manage global settings such as the calibration data of different sensors. There are methods to read settings from a configuration file, and it is possible to register properties such as procedures and fields, globally. These global properties can be read, written or called at runtime by the user, either programmatically or with simple script commands. Figure 11 shows the member classes `PropertyContainer`, `Section`, `Property` and all subclasses of it, and `Configurable`.

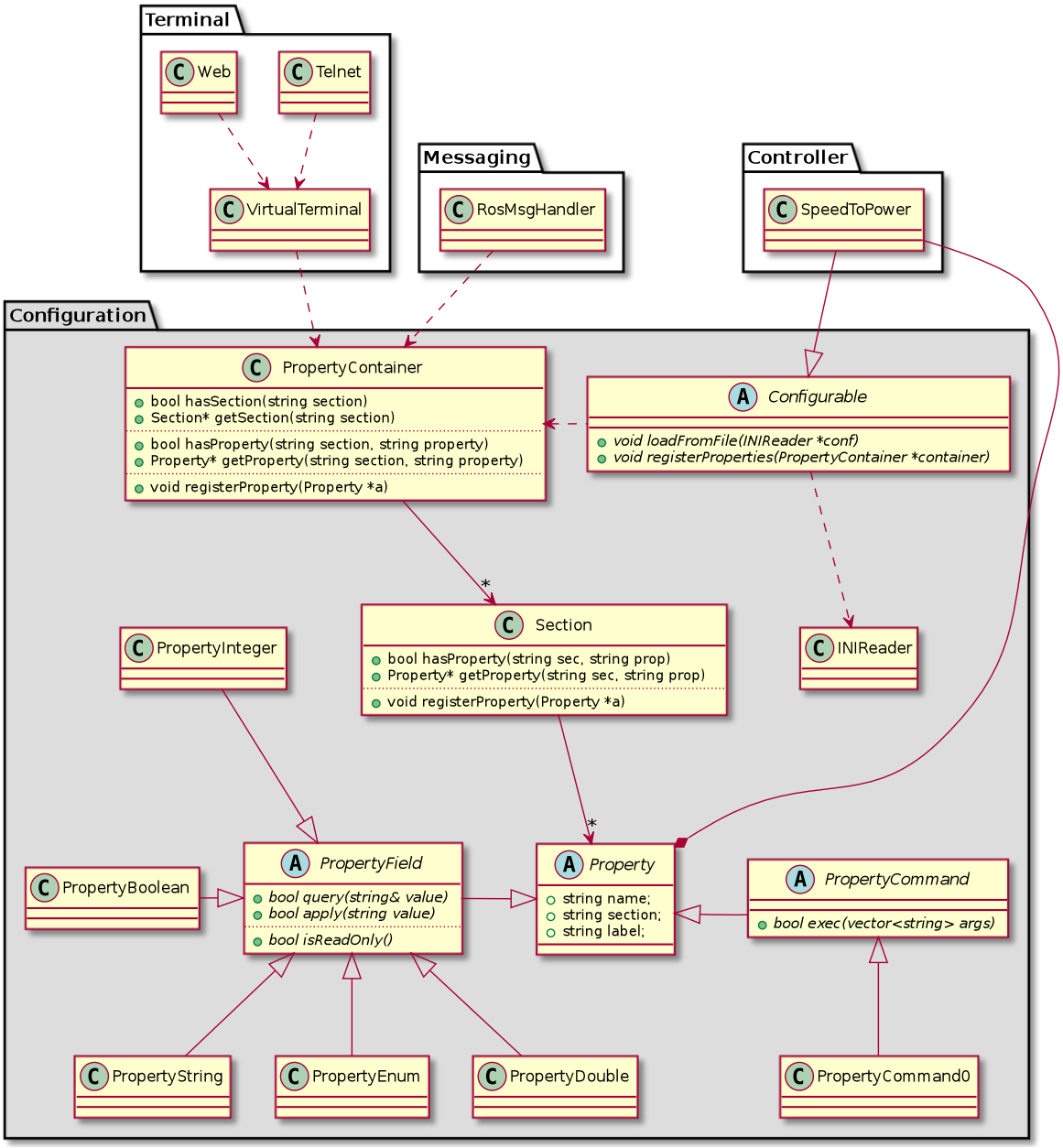


Figure 11: UML diagram of the `Configuration` package

PropertyContainer The core class in the configuration module is `PropertyContainer`. A property container owns a set of sections, which in turn contains one or more properties. The class `PropertyContainer` provides methods to register and query such properties of abstract type `Property`. Sections, implemented in class `Section`, can not be created directly, so there

are only methods to query them. A section is automatically created when a new property is going to be registered in.

Property There are two main types of properties. First, there is `PropertyField`. It is the superclass for fields. There exists multiple `PropertyField` implementations for several types like `boolean`, `string`, `double` and enumerated types. Field properties can be read only or readable and writeable. More types can be supported by sub-classing `PropertyField`.

The second property main type is `PropertyCommand`. It is a superclass for all procedures. A procedure accepts zero or more arguments, but is not returning any value. Currently, there exists only one implementation for the abstract class `PropertyCommand`. It is `PropertyCommand0`. This implementation covers procedures with no arguments. Procedures with multiple parameters can be supported by subclassing `PropertyCommand`.

Listing 1 shows an example how a property can be registered and accessed.

```
/*
 * Register a property of type double and name "Resolution" in section "HallSensor".
 * The possible values are between 10 and 200, and the stepsize is 0.01.
 * Lambda expressions are used to query and set field "resolution" that
 * actually stores the current resolution.
 */
container->registerProperty(new PropertyDouble("HallSensor", "Resolution",
  "Impulses per Meter", 10, 200, 0.01,
  [this]() { return resolution; },
  [this](double d) { resolution = d; }));

/* Query the property. NULL is returned if the property does not exist */
const Property* p = container->getProperty("HallSensor", "Resolution");

/* Cast the anonymous property to PropertyField and then query the value as string */
const PropertyField* field = dynamic_cast<const PropertyField*>(p);
if (field) {
  string s;
  if (field->query(s))
    print("Current Resolution: " + s);
  else
    print("Failed");
}
```

Listing 1: Register and access a `Property`

Configurable The abstract class `Configurable` acts as interface for all classes that publish properties and read from the configuration file. As configuration file we decided to use the well known INI file format[16]: The INI file format is an informal standard for configuration files for some platforms or software. INI files are simple text files with a basic structure composed of sections, properties and values. The structure is the same as it is being used already in `PropertyContainer`.

As an example, Figure 11 shows class `SpeedToPower`, which inherits from `Configurable`. Thus, `SpeedToPower` can access the configuration file and export properties over the property container. In fact, almost all classes from the `Driver` and `Controller` packages inherit from `Configurable`, and all of them are exporting multiple properties.

[16] *INI File.* Wikipedia. URL: http://en.wikipedia.org/wiki/INI_file. The INI file format is an informal standard for configuration files for some platforms or software. INI files are simple text files with a basic structure composed of sections, properties, and values.

2.2.2 Terminals

The previous section introduced class `PropertyContainer`. In this section we explain how it is used by the package `Terminal`. The terminal package provides simple access to the properties contained in `PropertyContainer` over Telnet[17] or a tiny web server[18]. Figure 12 shows the package `Terminal` with the three member classes `VirtualTerminal`, `Telnet` and `Web`.

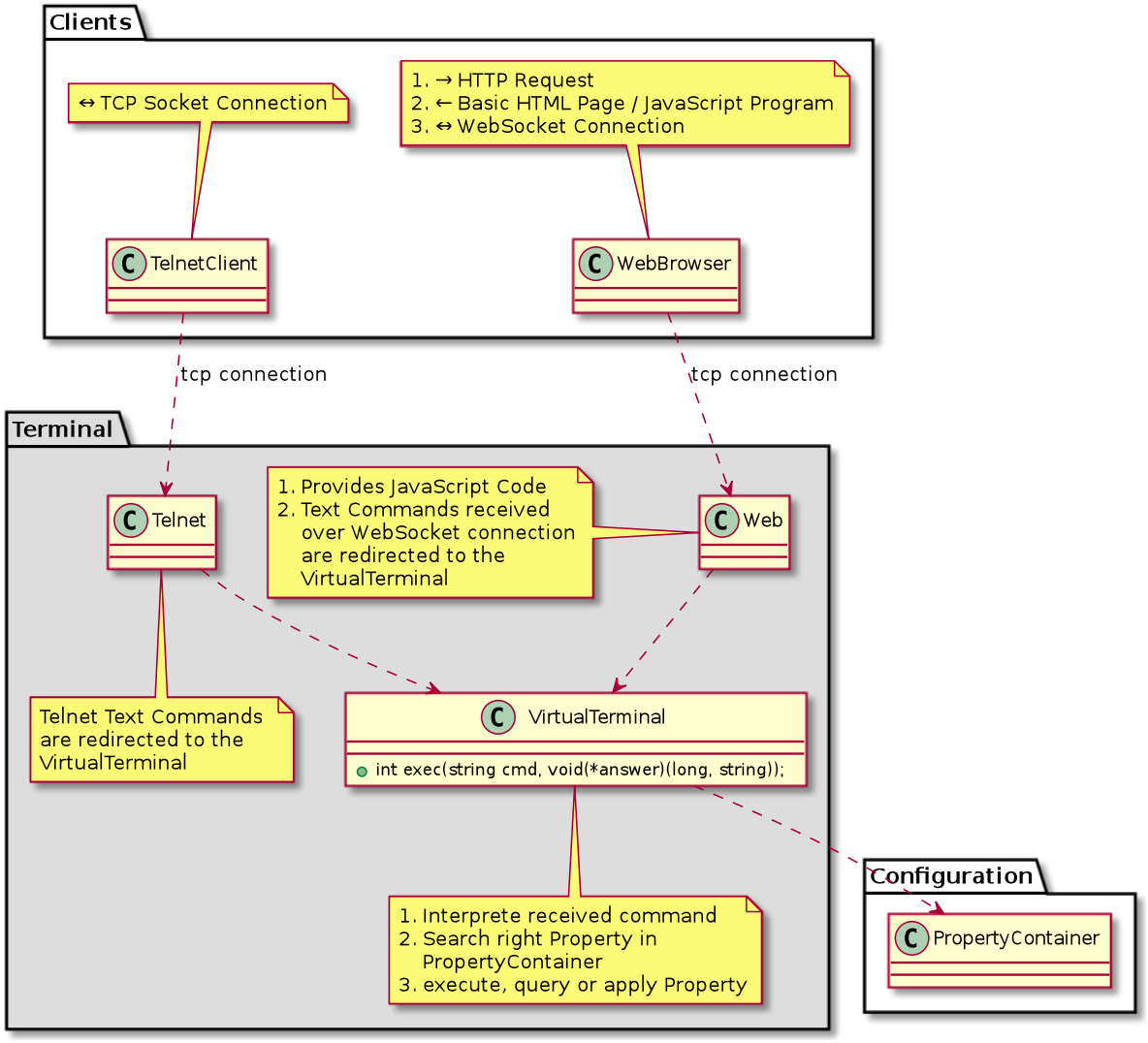


Figure 12: UML diagram of the `Terminal` package

VirtualTerminal The class `VirtualTerminal` uses the property container, to access its properties over string commands. As shown in Figure 12, `VirtualTerminal` has a public method `exec`. It takes a string command `cmd` as first parameter and a callback function `answer` as second one. After parsing the string command, the right `Property` is searched in `PropertyContainer` and then, depending on property main type, either `apply` or `query`

[17] *Telnet*. Wikipedia. URL: http://en.wikipedia.org/wiki/Telnet. Telnet is a network protocol used on the Internet or local area networks to provide a bidirectional interactive text-oriented communication facility using a virtual terminal connection.

[18] *Web server*. Wikipedia. URL: http://en.wikipedia.org/wiki/Web_server. A web server is a computer system that processes requests via HTTP, the basic network protocol used to distribute information on the World Wide Web.

in case of `PropertyField` or `exec` in case of `PropertyCommand` is called. If there are any results, then the callback function `answer` is invoked once per result line.

The first argument of the callback function `answer` is the command identifier, the second one the message itself. The command identifier is an optional prefix of type integer for each command. With that identifier it is possible to create a relation between the sent command and the asynchronous replies. The command and reply syntax is explained in Figures 13 and 14.

Telnet The `Telnet` class implements a basic Telnet server. The implementation is straight forward: A TCP server socket listens for incoming connections. After a client connection is established, string commands are accepted and directly forwarded without modification to the `VirtualTerminal` instance. Also the answers from the virtual terminal are sent back over the TCP socket to the client without any modifications.

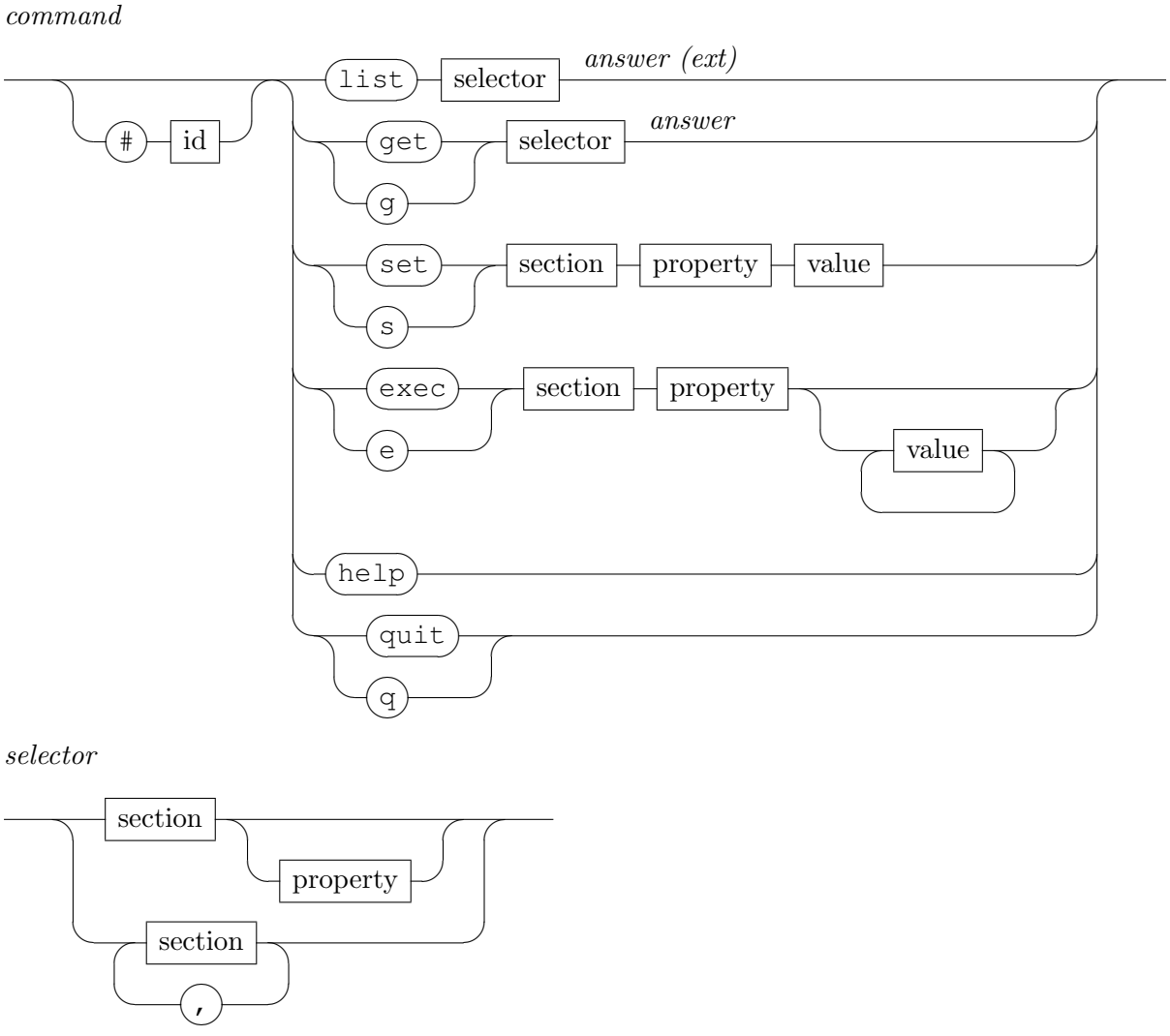


Figure 13: Command syntax of `VirtualTerminal` interpreter

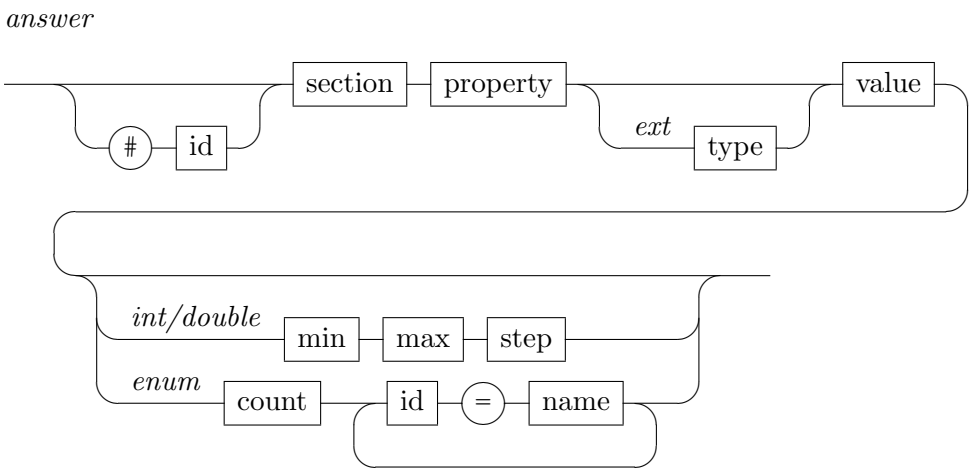


Figure 14: Reply syntax of `VirtualTerminal` interpreter

Web As a simple but expandable solution to bring a graphical user interface onto different devices, we opted for a HTML5 web application. It can run on a normal browser without any special client software. Therefore we implemented a very simple web server into our controller software. For the web server, we decided to use POCO[19], a collection of open source C++ class libraries and frameworks for building network- and internet-based applications that run on desktop, server, mobile and embedded systems.

The web server serves only a very small HTML5 page with a basic skeleton of HTML[20] [28] tags. It also defines an embedded JavaScript variable that stores a list of possible applications available on the server. From this variable the client can read what other applications can be served by the web server. Current applications are a *Joystick App* to remote control the walker, a *Properties App* to read and set all properties and a *Monitor App* to monitor speed and power data from assist mode. The applications are provided as JavaScript file together with a CSS file for the style. Developers can provide their own JavaScript and CSS files to extend the list of applications. Figure 15 shows a screenshot of the *Properties App* application.

The JavaScript client application can open a WebSocket[21] connection for a full-duplex communication with the BeagleBone controller. The same protocol must be used, as described in the previous paragraphs **VirtualTerminal** and **Telnet**. For simplicity, the web server provides a JavaScript file that offers a set of helper functions to easily communicate via WebSocket with the controller.

[19] *POCO C++ Libraries*. POCO. URL: http://pocoproject.org/. Modern, powerful open source C++ class libraries and frameworks for building network- and internet-based applications that run on desktop, server, mobile and embedded systems.

[20] *HTML5*. Wikipedia. URL: http://en.wikipedia.org/wiki/HTML5. HTML5 is a core technology markup language of the Internet used for structuring and presenting content for the World Wide Web.

[21] *WebSocket*. Wikipedia. URL: http://en.wikipedia.org/wiki/WebSocket. WebSocket is a protocol providing full-duplex communications channels over a single TCP connection. The WebSocket protocol was standardized by the IETF as RFC 6455 in 2011.

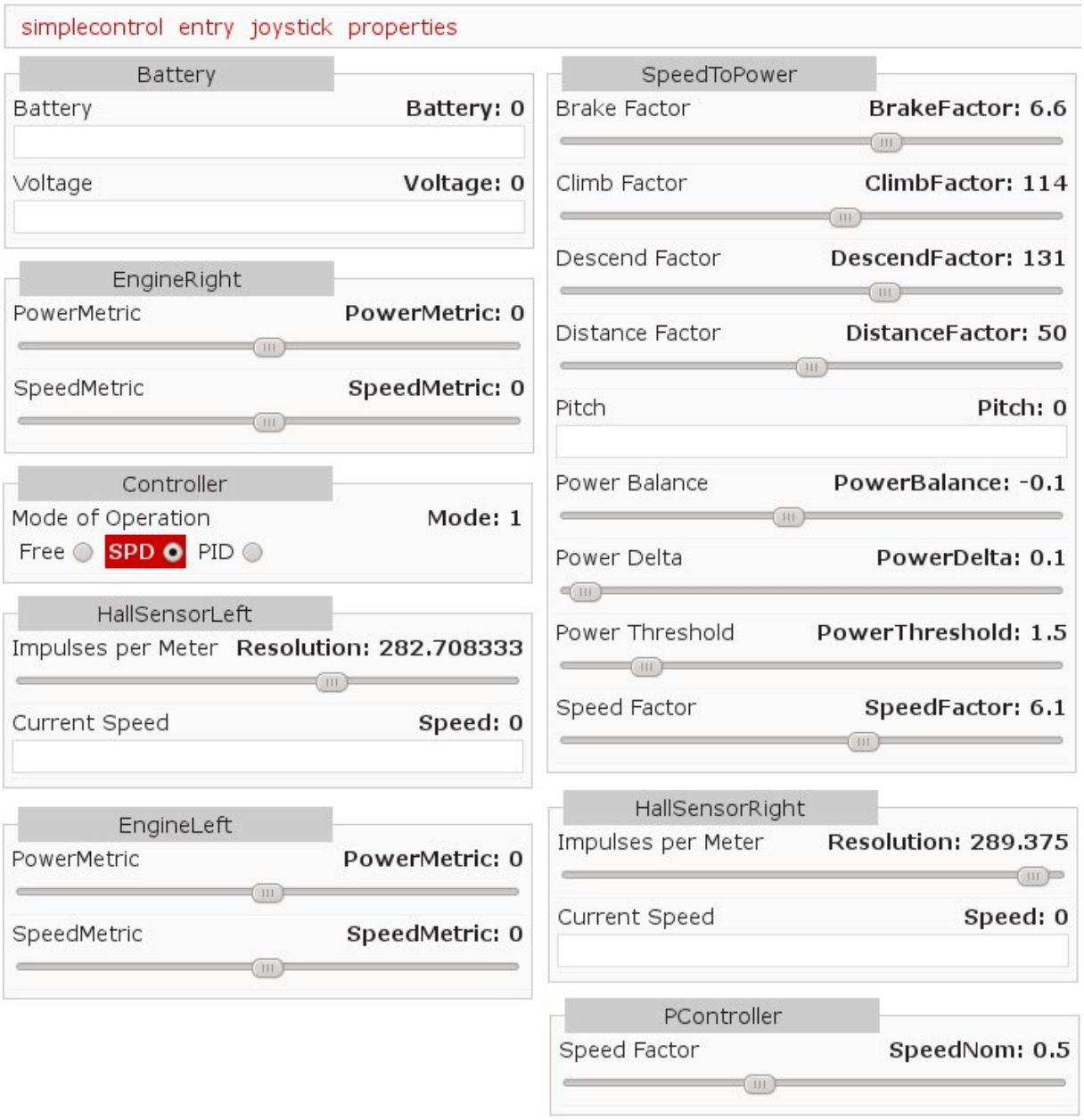


Figure 15: Web user interface (Properties App)

2.2.3 ROS

The controller software supports also the Robot Operating System (ROS) [35] in the version of Indigo Igloo[22]. In short, ROS is a collection of software modules for robot development. The SmartWalker controller acts as ROS node, which can be integrated into a ROS cluster.

Figure 16 shows the `ROS` package with the classes `MsgHandler` and its two subclasses `RosMsgHandler` and `NullMsgHandler`. The ROS code is implemented in the class `RosMsgHandler` which inherits from the abstract class `MsgHandler`. The class `NullMsgHandler` is a dummy or null implementation of `MsgHandler` and can be instantiated in place of `RosMsgHandler` to keep ROS disabled.

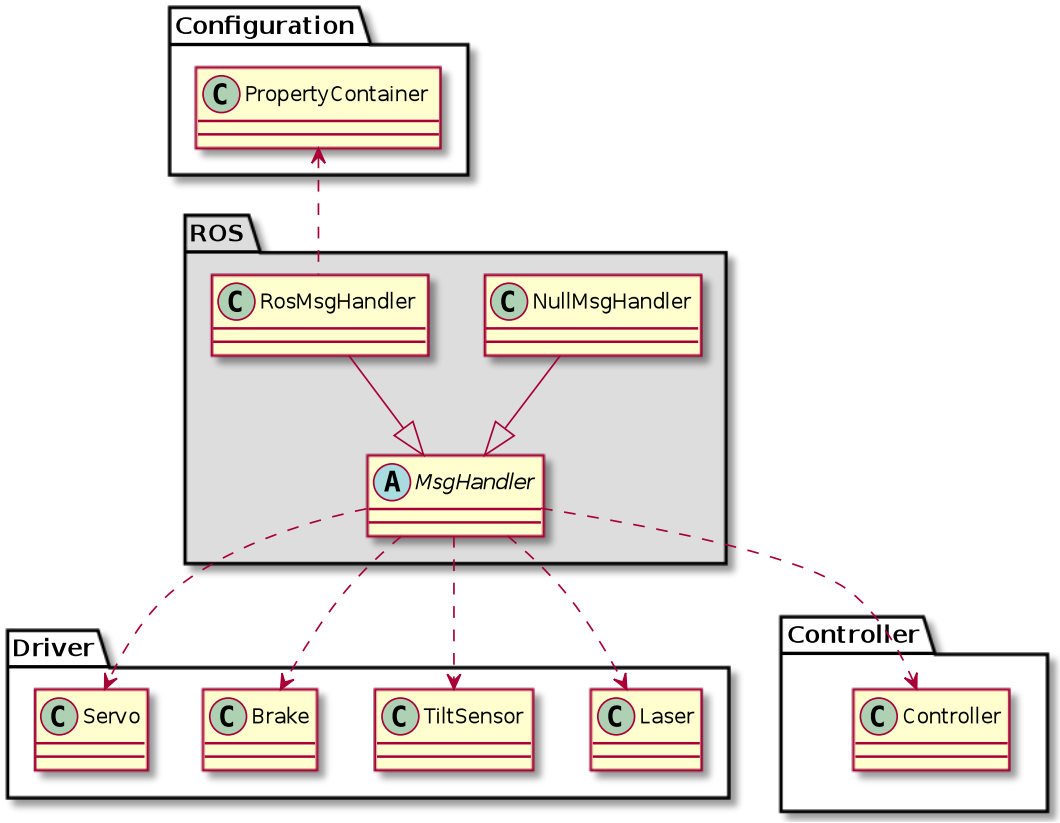


Figure 16: UML diagram of the `ROS` package

ROS allows developers to access all actuators and sensors over the advertised or published ROS topics. With the custom ROS message `SetProperty`, one can also set or execute properties managed by the property container. Table 1 shows all subscribed topics, their message types and a short description. Table 2 shows the same for all advertised topics.

[22] *ROS Indigo Igloo*. ROS. URL: http://wiki.ros.org/indigo. ROS Indigo Igloo is the eighth ROS distribution release and was released July 22nd, 2014.

Topic	Type	Description
/desired_speed	`geometry_msgs::Twist`	set the speed of the walker
/desired_lspeed	`std_msgs::Float32`	set the speed of the left wheel in ms^{-1} (SpeedMetric)
/desired_rspeed	`std_msgs::Float32`	set the speed of the right wheel in ms^{-1} (SpeedMetric)
/desired_lpower	`std_msgs::Float32`	set the power of the left wheel (PowerMetric)
/desired_rpower	`std_msgs::Float32`	set the power of the right wheel (PowerMetric)
/primesense_angle	`std_msgs::Float32`	set the angle of the PrimeSense servo °
/set_property	`roboscoop_ros::SetProperty`	set or execute a property

Table 1: Subscribed ROS topics

Topic	Type	Description
/current_speed	`geometry_msgs::Twist`	speed of the walker (twist)
/current_position	`geometry_msgs::PoseStamped`	pose
/nav_odometry	`nav_msgs::Odometry`	odometry data (pose and twist)
/step_lwheel	`std_msgs::Int16`	hall effect sensor tics of the left wheel
/step_rwheel	`std_msgs::Int64`	hall effect sensor tics of the right wheel
/counter_lwheel	`std_msgs::Int64`	hall effect sensor tic counter of the left wheel
/counter_rwheel	`std_msgs::Int64`	hall effect sensor tic counter of the right wheel
/laserScan	`sensor_msgs::LaserScan`	LiDAR data
/battery_state	`std_msgs::Float32`	state of the battery between 0 and 1
/lbrake	`std_msgs::Bool`	state of the left brake
/rbrake	`std_msgs::Bool`	state of the right brake

Table 2: Advertised ROS topics

2.2.4 Position

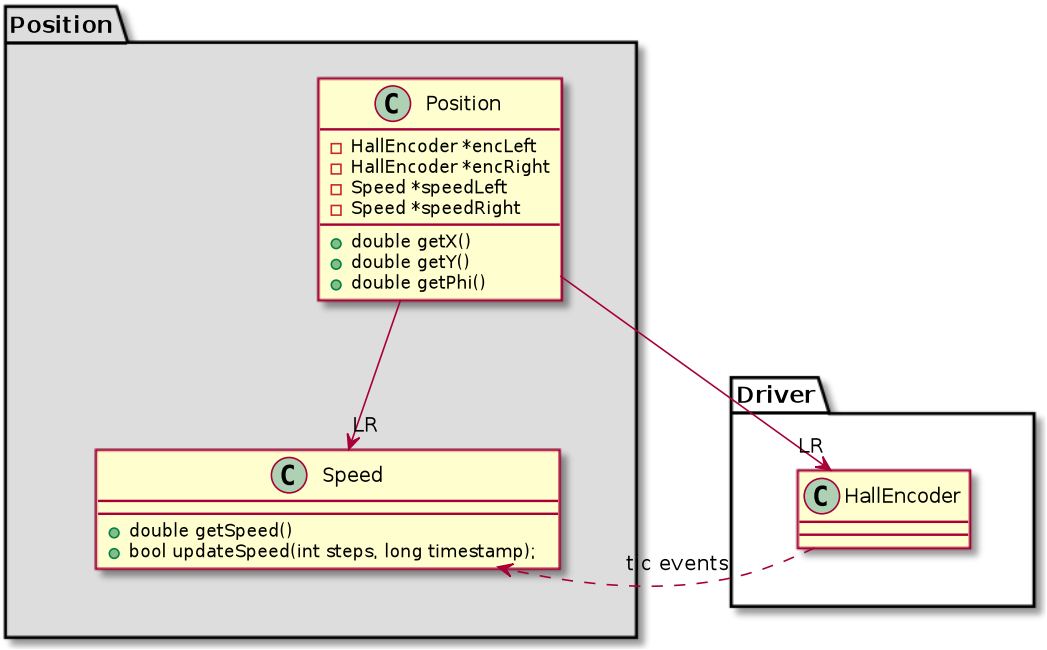


Figure 17: UML diagram of the `Position` package

The `Position` package consists of just two classes, the `Speed` and `Position` class (See Figure 17). The functionality of these classes is easily explained. For both hall effect sensors, one per motorized wheel, there is a driver instance `HallEncoder`, which sends timestamped tic messages to the `Speed` class. The frequency of these tics is linearly dependant to rotation speed of the wheel. Thus the speed class can compute the rotation speed out of the received tic events. In turn, the class `Position` uses both speed instances in order to calculate the movement and therefore the position of the walker.

2.2.5 Threading

This section explains the used concurrency model. SCOOP [27] is Simple Concurrent Object-Oriented Programming. It is a model for concurrent computation, whose objectives are to abstract the notion of concurrency to a level above the tools and techniques that are currently available in the target concurrency environment, such as threads, mutex and semaphores.

In SCOOP it is possible to declare certain object types as `separate`. A separate instance of an object may be handled by another processor, it is an abstract notion of an autonomous thread. Principles of Design by Contract [25] are used to synchronize the access to shared resources.

SCOOP is part of Eiffel Studio[23], which is not available for ARMhf so far. Therefore it is not possible to use SCOOP on embedded devices as BeagleBone. But our design of running classes on their own thread is similar to separated instances in SCOOP.

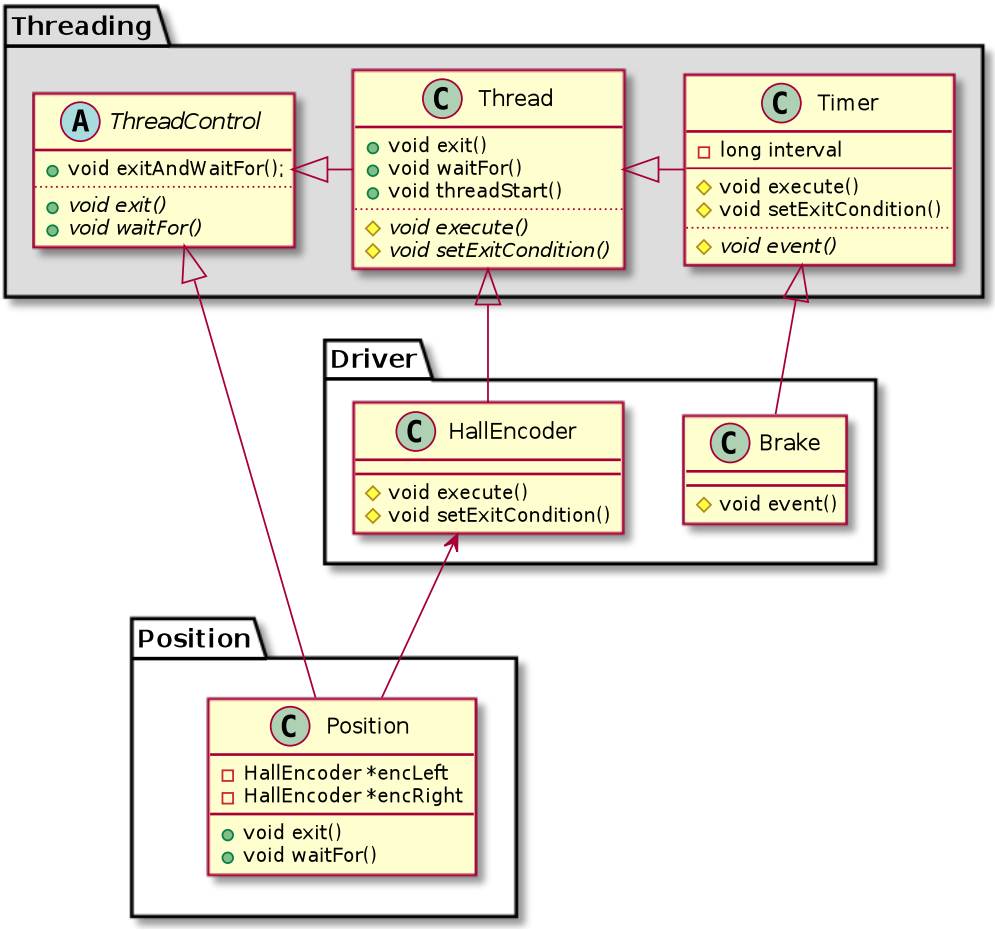


Figure 18: UML diagram of the `Threading` package

Almost all classes of the `Controller`, `ROS`, `Position`, and `Driver` packages inherit from one of the classes contained in the `Threading` package. Figure 18 shows the member classes `ThreadControl`, `Thread` and `Timer`.

The class `ThreadControl` is a super class for all those classes which own one or more threads or which run itself separately. This does not mean that the class itself is separated. With the method `exit`, the caller signals the callee that it must stop at the earliest opportunity, including

[23] *Eiffel Software*. Eiffel Software. URL: https://www.eiffel.com/.

the owned threads. With `waitFor`, a caller can join the callee and return only when the joined thread is terminated.

A class inheriting from `Thread` creates its own thread. After instantiation, the method `execute` is called. The class instance is valid as long as the process does not return from this execute method.

The third threading class is `Timer`. It inherits from `Thread` and implements the abstract method `execute`. Additionally, it defines the abstract method `void event()`. This method is called within a fixed frequency. The frequency is provided as parameter to the constructor of the class `Timer`.

In Figure 18, the class `Position` is used as an example that inherits from `ThreadControl`. Therefore Position is not located on its own thread, but it owns two instances of `HallEncoder` which themselves inherits from `Thread`. The `Brake` driver polls the state of the brake regularly, so this class inherits from `Timer`.

2.2.6 Drivers

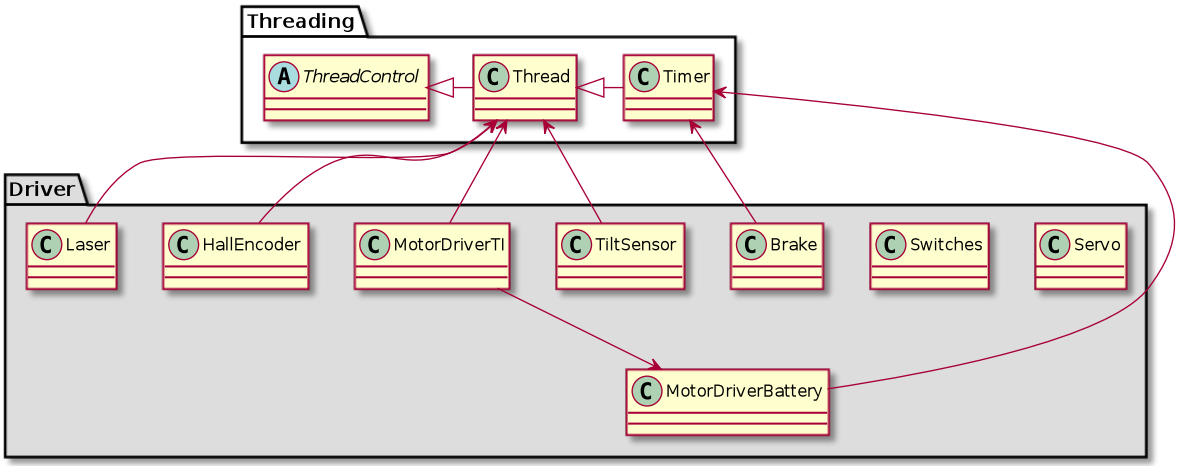


Figure 19: UML diagram of the `Drivers` package

The `Drivers` package has classes that manage and provide access to all hardware devices. The classes are listed in Table 3 and shown Figure 19.

Class	Device	Sections
`Laser`	XV-11 Laser Scanner	2.1.4, 3
`HallEncoder`	XV-11 Laser Scanner	2.1.2, 2.2.4
`MotorDriverTI`	Stellaris LM3S8971 BLDC Motor Control Board	2.1.2, 5
`TiltSensor`	Pewatron PEI-Z100-AL-232-1 360°	2.1.5, 5
`Brake`	Tektro EL550-RS and GPIO of motherboard	2.1.3, 2.1.1, 5
`Switches`	GPIO of motherboard	2.1.1, 5
`Servo`	Servo Motor for PrimeSense	2.1.6

Table 3: Driver classes and devices

3 Leg Detection

One of the main objectives of this project is to track the user's leg movement. Using this information in assist mode, the velocity can be adjusted according to the distance between the walker and the person. If the distance increases between the person and the rollator, the walker is probably too fast, or too slow, when the distance decreases. If there are no detected legs, the controller can assume that there is no one behind the walker.

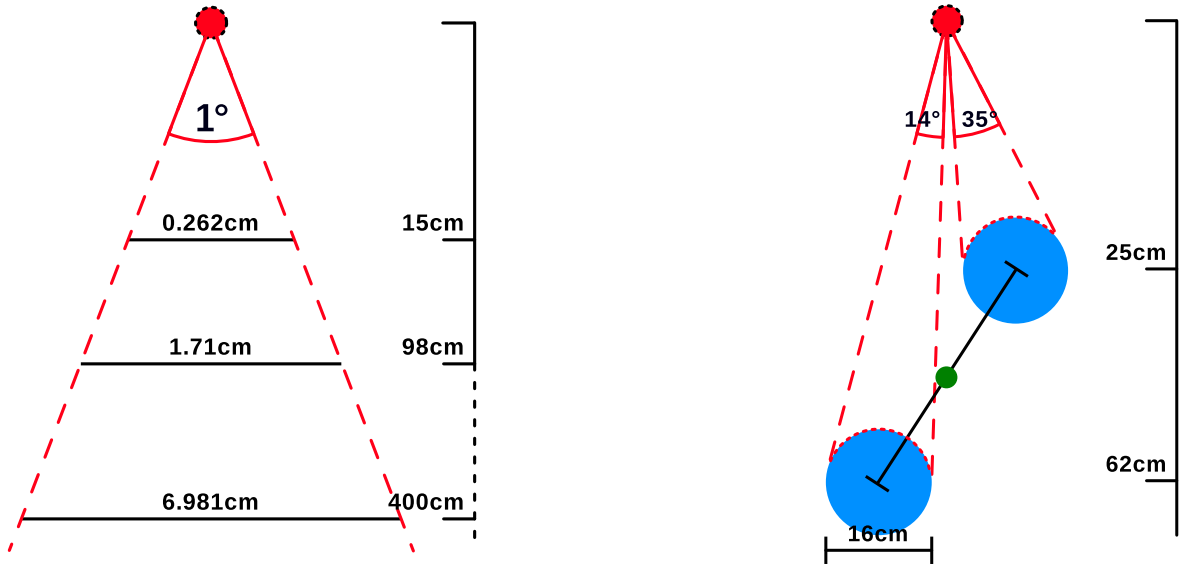
3.1 LiDAR Capabilities

The LiDAR sensor is used to scan the area around the walker. The XV-11 scanner (Section 2.1.4) has a resolution of one ray per degree. The rotation speed is at $4Hz$, therefore the latency between two consecutive scans is $694\mu s$, and a full 360° scan takes around $250ms$.

The distance between two consecutive scanned points is around $0.262cm$, when the measured object is $15cm$ away from the laser scanner. If this distance between laser and object grows up to $4m$, then the measured distance between two consecutive points grows to $6.981cm$. Figure 20a shows to width of the view range between two consecutive laser rays in distances of $15cm$, $98cm$ and $400cm$.

From the evaluation of walking distances (See Figure 38) we know that people are walking in a distance between $40cm$ and $60cm$ to the rollator. Figure 20b outlines the case, where a person is at a distance of $43cm$ behind the walker. The blue circles are the legs, the green circle the body centre.

The number of rays that may hit a leg of $16cm$ in radius, is between 14 and 35 in this case because the spanned angle between the two consecutive rays is 14° for the left, and 35° for the right leg. But this is the optimal case. Depending on the quality of reflection, it is possible that some measurements fail and therefore less data points are available for processing.



(a) LIDAR resolution

(b) Rays per leg

Figure 20: XV-11 LiDAR capabilities

3.2 Detection

Kim, Chung, and Yoo [20] proposed to distinguish legs from other objects by characteristics that a leg cannot have with high probability. They defined an area in front of the robot, where the tracked person have to go in, and they expect the target person to wear long trousers. In the first step, the authors use an incremental method to exclude objects with linear features exceeding a certain length. After clustering they filter against the defined characteristics of a leg.

Following a human by tracking its legs, as proposed by Kim, Chung, and Yoo [20], requires a much more advanced technique in detection and tracking of legs than what is needed for this project. Figure 21 shows the individual steps for detecting the legs. In the first step we read the data points from the laser scanner and transform the coordinates from the polar to the Cartesian coordinate system. Then, data points outside of the walking area are filtered and EM clustering is applied to the remaining cloud. Finally we do a simple validation of the found clusters.

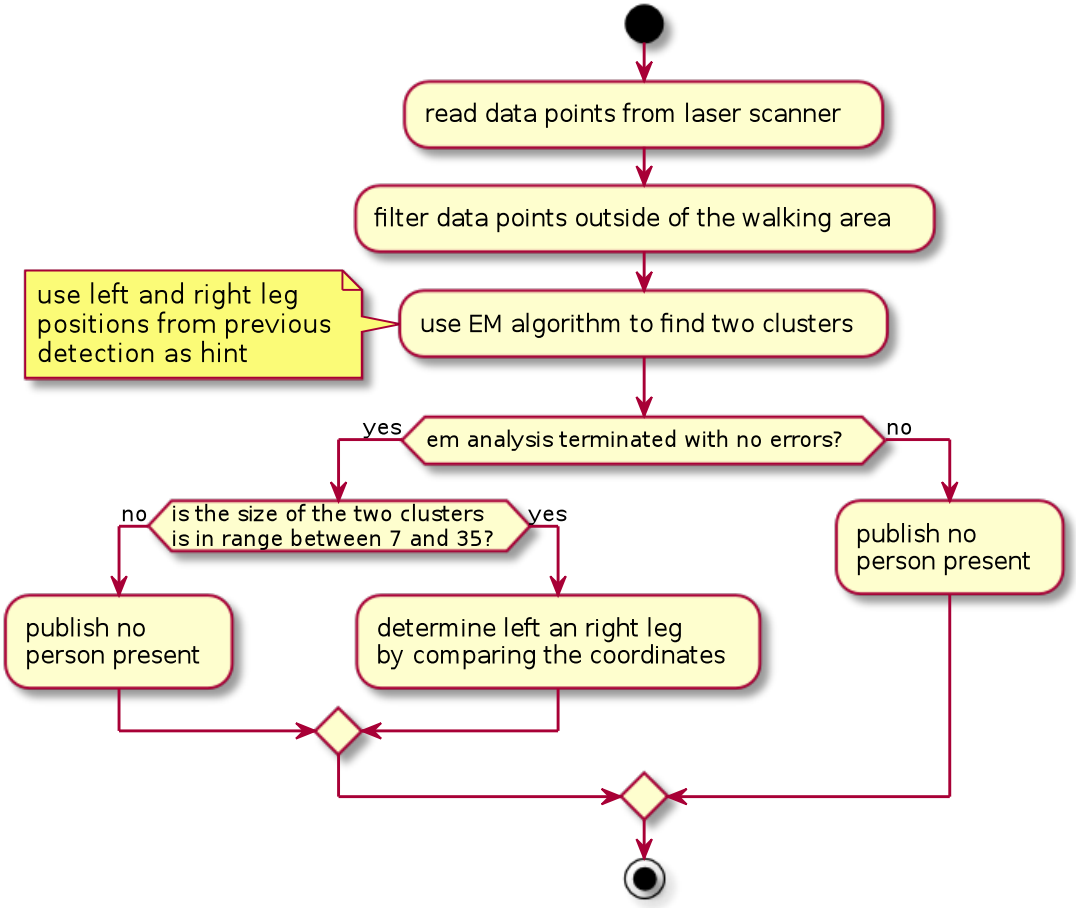
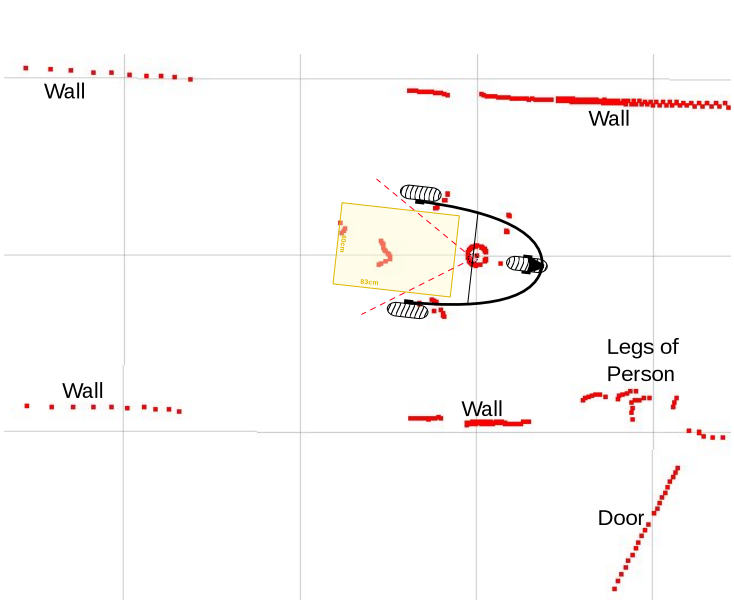
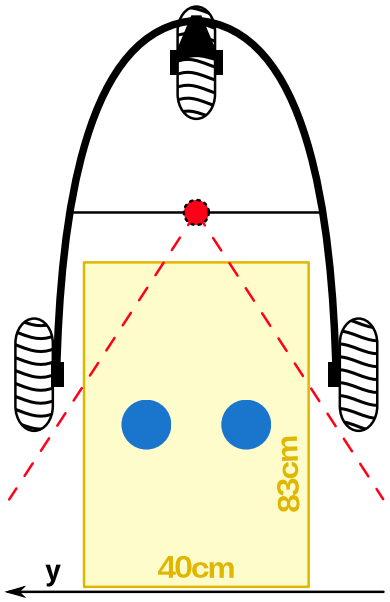


Figure 21: Leg detection algorithm

Filtering In the second step we filter all data points that belong to the scene around the walker. Figure 22a shows the walker within the scene and Figure 22b the walker with the defined walking are. We expect the person to stay or walk inside this pre-defined walking area. Objects outside of this area are ignored. We expect a person to wear trousers and no other objects inside the detection area, besides exactly two legs.



(a) Snapshot of laser scan data with walker



(b) Walking area behind the walker with initial leg positions

Figure 22: Filter data points outside of the walking area

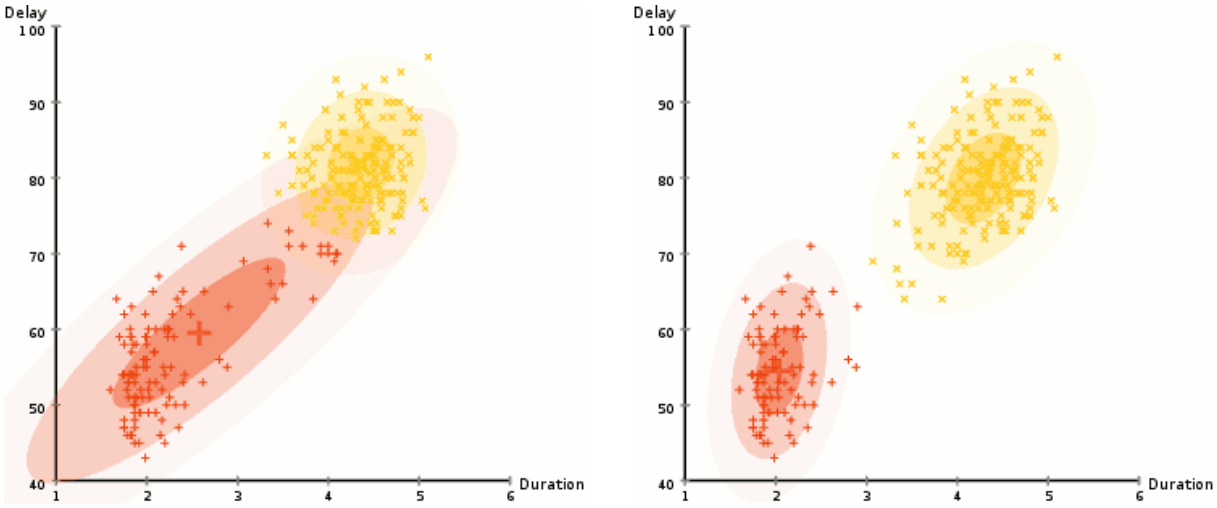
Clustering We assume exactly two legs or none inside the leg area, therefore there are also two clouds of data points, one for each leg. With clustering[24] [44] we assign each data point either to the left or to the right leg. The two clouds are always distinct from each other and it may never happen that one cloud is contained inside the other one. Therefore we can use the Expectation–maximization algorithm (EM) for the cluster analysis [10]. The EM algorithm is an iterative method for finding maximum likelihood estimates of parameters.

The EM algorithm expects the point cloud and an initial guess of the cluster positions and the search radius as input. The output of the EM algorithm are the computed cluster centres. Because we are interested in two clusters, we have to guess two positions. Our algorithm takes either the leg positions of the last cycle as input, or if this information is too old or not available, then we use the initial leg positions. The initial leg positions are the blue circles in Figure 22b.

If there are errors during clustering, as floating point exceptions, then we assume that there is no person present.

Figure 23a shows two clouds (dots) with two guessed positions (big cross) and two search areas (circles). The EM algorithm computes the clusters centres as shown in Figure 23b

[24] *Cluster analysis*. Wikipedia. URL: http://en.wikipedia.org/wiki/Cluster_analysis. Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters).



(a) Initial situation

(b) Final situation

Figure 23: EM Clustering [13]

Validation After clustering we know the cluster positions and the size. We know from Section 3.1 that a leg consist of up to 35 scan points. If a cluster has less than 7 or more than 35 scan points, then it is not a leg. In such cases we drop the clusters and assume that no person is present.

If the cluster size is in the valid range, we determine which cluster relates to which leg. This is done by comparing the y component of the coordinate. The cluster with the smaller y component represents the left leg, the one with the bigger y the right leg.

3.3 Distance Computation

From the leg detection we know the position of the left leg $\vec{gl}$ and right leg $\vec{gr}$. The distance d between the walker and the person is one of the inputs for the speed to power controller (Section 5.3). This distance is calculated as follows: First we compute the body centre c and then the distance d between it and the laser scanner. The laser scanner is located at the origin of the coordinate system.

The leg detection algorithm delivers 4 distances d per second because the LiDAR rotates at $4Hz$. These distances are stored in round robin cache t of size n. Size n is heuristically set to 360, which means that oldest cache entry is 90 seconds old. The mean distance $\widetilde{d}$ is the mean value over all elements of cache t.

$$\vec{c} = \frac{\vec{gl} + \vec{gr}}{2}$$

$$d = \sqrt{\vec{c_x}^2 + \vec{c_y}^2}$$

$$\widetilde{d} = \frac{1}{n} \sum_{i=0}^{n-1} t_i$$

3.4 Implementation

In this section we explain in short the implementation of the em algorithm. The EM algorithm repeats the following two-step procedure to refine the result.

- **E-step** Apply a Kalman filter to obtain updated estimates [19]
- **M-step** Use the smoothed state estimates within maximum-likelihood calculations to obtain updated parameter estimates

For our EM algorithm implementation we have taken ideas from Jormungand[25]. This algorithm uses elements from the Linear Discriminant Analysis (LDA) for both steps [24]. We extended the code to support a second termination criterion. The first criterion is the maximum number of iterations, the new criterion is a threshold. If changes from one step to the next are below this threshold, the loop terminates. We also specialized the code that it supports exactly two clusters, this way a lot of unnecessary code could be removed. The code is optimized for spatial and temporal locality and small loops are unrolled. With these optimizations it is possible to run leg detection completely on BeagleBone (Listing 3 of Appendix A shows the implemented code).

[25] Jormungand Chris. *EM Clustering Algorithm*. URL: http://jormungand.net/projects/misc/em/.

4 Environment Tracking

Due to uncertainty and error, the position, which is tracked by the class `Position` with information from the hall effect sensors, drifts away more and more over time from the actual position. Outside, that can be corrected with the help of a GPS sensor. But inside, this option does not exist. There exists a class of algorithms that help tracking objects and environments, called Iterative Closest Point (ICP) algorithms [2][6]. ICP can be used to localize robots and achieve optimal path planning.

Feng and Milios [23] present two algorithms for self-localization. The first algorithm is based on matching tangent lines defined on two scans. The second algorithm algorithm is based on iterative least-squares solutions using point to point correspondences, similar to the ICP algorithm in [2][6].

Odometry data alone is not sufficient to solve the localization problem. Therefore it is recommended to use visual sensors to match environmental features. SmartWalker has two visual sensors, the PrimeSense and the XV-11 LiDAR, but the perspective of the PrimeSense is small. We can reuse the 360° image data of the LiDAR also for environmental tracking. Feng and Milios [23] suggest to use a 2D laser range scanner for pose estimation.

4.1 CSM

Naive ICP implementations are computational intensive. BeagleBone has limited computational capacities, therefore we have to rely on a highly optimized implementation.

Censi [5] describes PLICP, an ICP variant that uses a point-to-line metric, and an exact closed-form for minimizing such metric. The Canonical Scan Matcher (CSM)[26] is an implementation of PLICP.

The integration of the CSM library into our controller was possible. The algorithm requires the approximate current position (odometry) and the laser range data as input. The approximate current position is taken from the `Position` instance, while the laser range data is delivered as raw data directly from the XV-11 driver `Laser` to the CSM code.

4.2 Evaluation

We conducted a short feasibility study in order to examine the use of the XV-11 LiDAR for environment tracking.

4.2.1 Setup

For the evaluation we defined two tracks: Figure 24 shows a red and a green track passing through the same rooms. One track is the opposite to the other. The red track starts in a narrow corridor and leads first $10m$ straight ahead. Then there is a 90° right corner, which is followed by a $4m$ hallway again. Then the hallway opens into the entrance area. Here we make a 180° turn. The green track on the map goes into the opposite direction, compared to the red one.

[26] *CSM*. Massachusetts Institute of Technology. URL: http://censi.mit.edu/software/csm/. The C(anonical) Scan Matcher (CSM) is a pure C implementation of a very fast variation of ICP using a point-to-line metric optimized for range-finder scan matching.

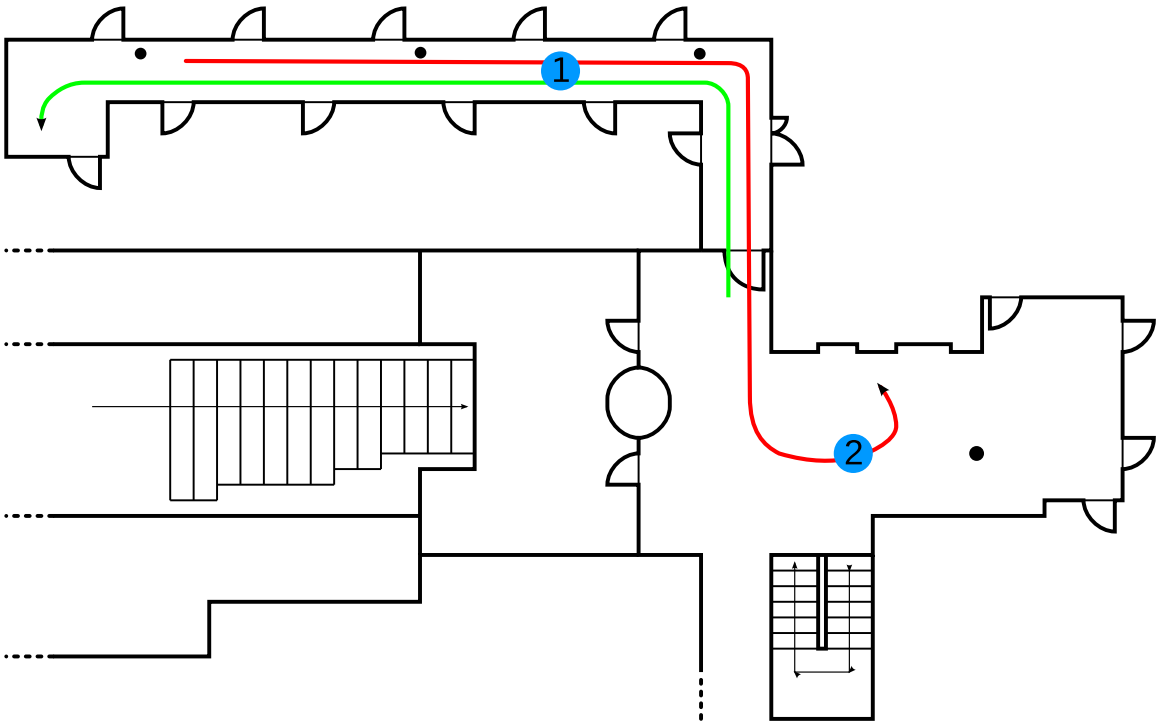


Figure 24: Map for CSM evaluation

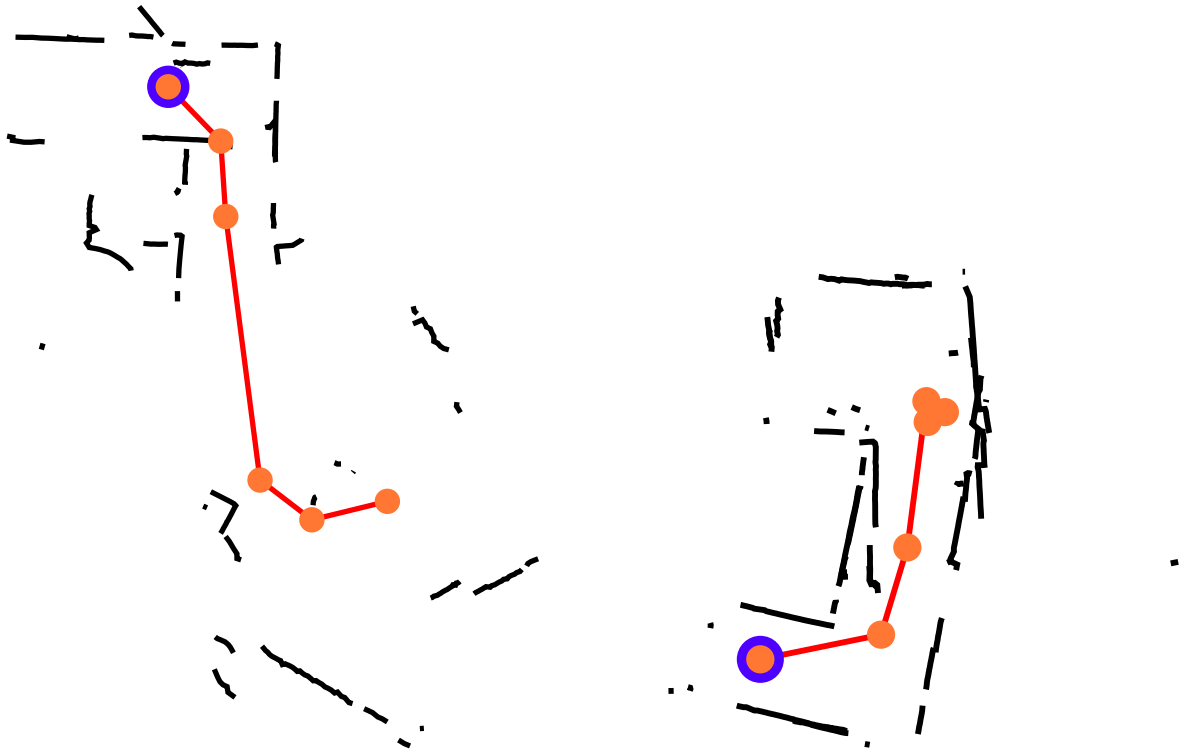
4.2.2 Results

The movement around the corners is detected and tracked well. But the straight, forward moving inside the corridor is not detected at all. Figure 25 shows the traced environment, where the orange line is the traced moving. Figure 25a should match the red track, Figure 25b the green one. In these figures one can see that the corners are in direct succession and the straight corridor is not detected. The hallway has too few rough features or the density of the XV-11 is too small to measure fine grained features.

Figure 26 visualises two snapshots of recorded laser scan data. One snapshot is taken at point 1 inside the corridor, the other one at point 2 inside the entrance area. The red triangles of Figure 26a are the dead angles, caused by the wheels. This dead area is huge and therefore, a big part of the environment cannot be seen from the point of view of the laser scanner.

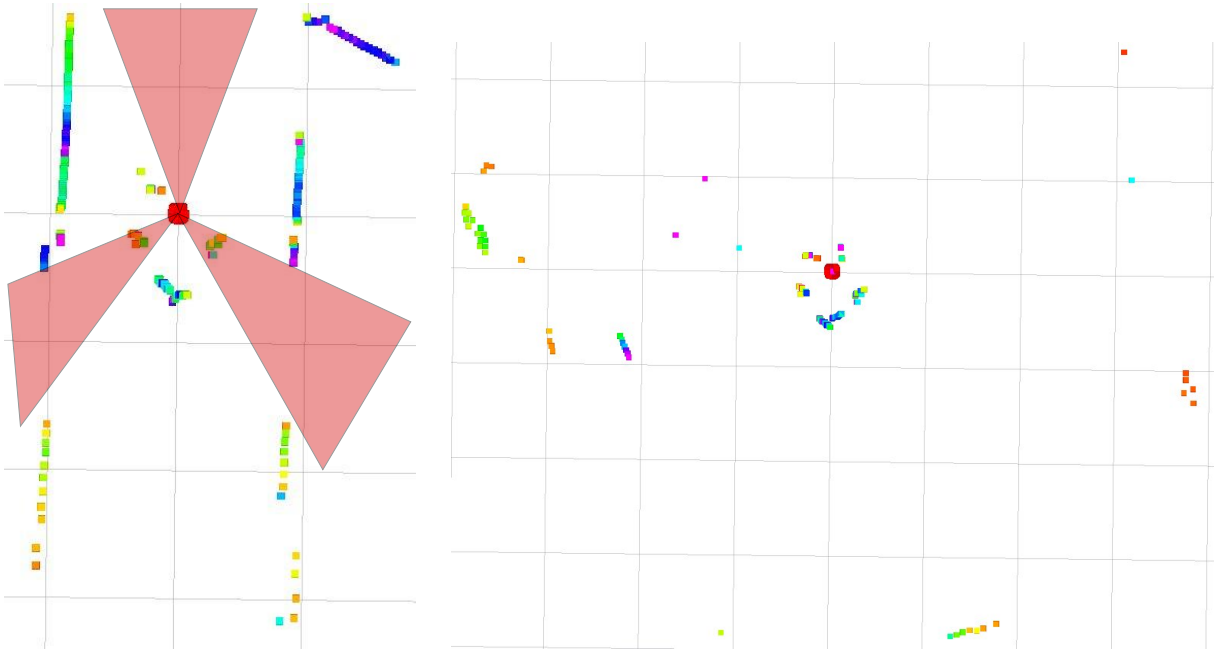
In Figure 26b, there is almost nothing to see. The dimension of the entrance area is simply too big for the XV-11. Objects beyond $4m$ are only partly detected and the distance between two consecutive scan points is around $7cm$. With a distance of $7cm$ between two scan points it is not possible to detect small features.

This leads to the conclusion that environment tracking with the XV-11 is possible only in tiny areas with an extent of less than $2m$ and a full 360° sight radius.



(a) Traced red track (b) Traced green track

Figure 25: Traced environment with CSM algorithm



(a) Inside corridor (Point 1) (b) Entrance area (Point 2)

Figure 26: Visualization of laser scan data during CSM evaluation

5 SmartWalker Controller

The controller code is the heart of the robot. It receives data from the sensors and users, interprets them, and then controls the motors. The controller package (Figure 27) has two built in controller modes. The proportional-integral-derivative controller (PID) is implemented by the class `PController` and is explained in Section 5.2, and the speed to power controller (STP), implemented by class `SpeedToPower` (Introduced in Section 5.3). Both controller modes are using the class `WheelController` as safety authority between them and the motor driver. Section 5.1 explains the functionality of class `WheelController`.

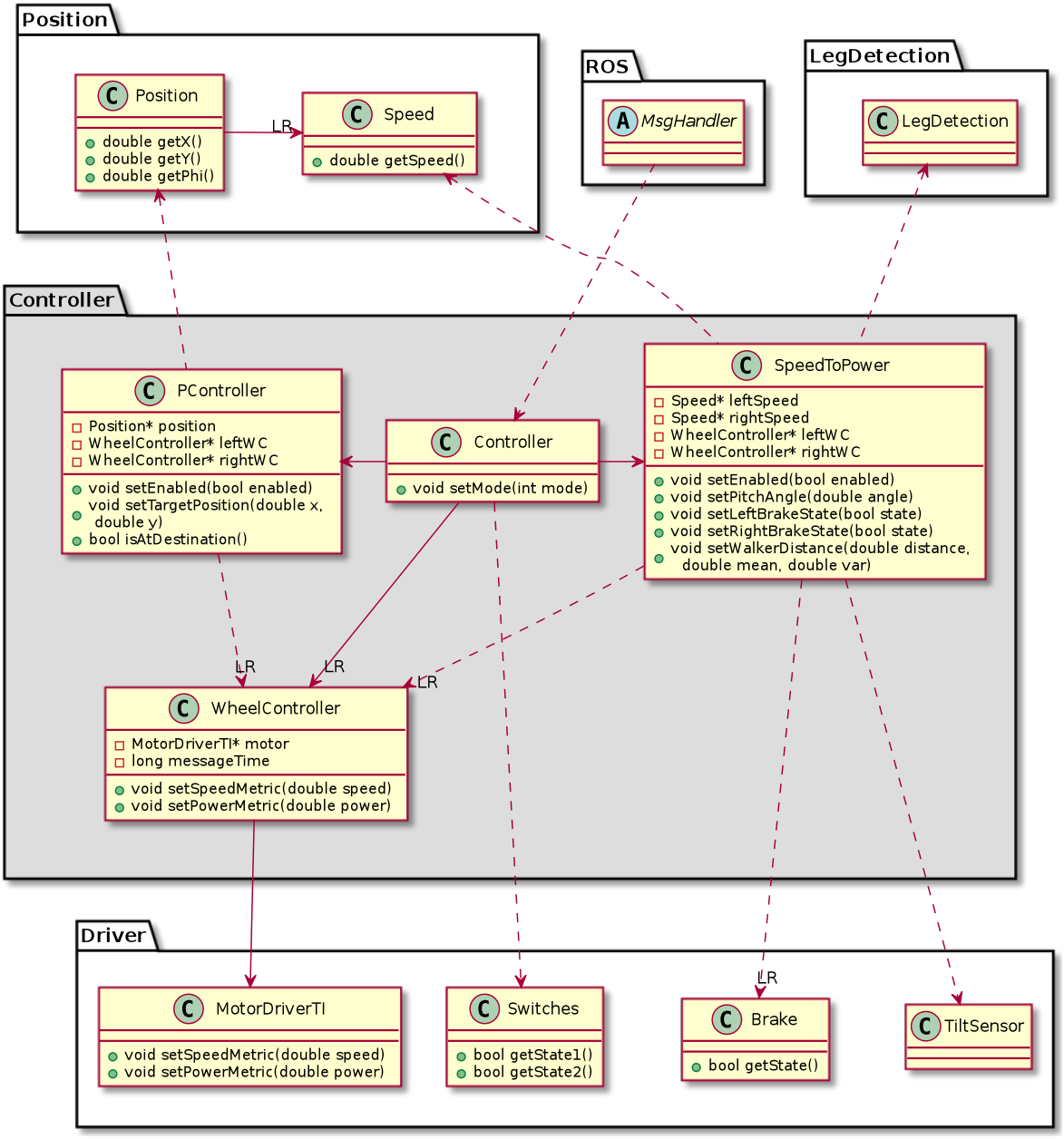


Figure 27: UML diagram of the `Controller` package

The class `Controller` is the manager of the controller modes. It can either set one of the them active or disable both. If both modes are disabled, the SmartWalker can be controlled either over terminal commands (Section 2.2.2) or by ROS (Section 2.2.3).

The controller manager reads also the status of the hardware switch, which is soldered on the controller board. With this switch it is possible to keep both modes deactivated. It is no longer possible to enable one of the modes programmatically. This can be useful if developers plan to use ROS only.

5.1 Wheel Controller

The class `WheelController` is a safety authority between an active controller mode and the motor driver. If the wheel controller does not receive speed or power messages or commands regularly, the engines are stopped automatically. This mechanism is important because it prevents the wheels from continuously turning. One reason for this may be that a parent controller hangs, or that ROS messages are not received due to an interruption of the network connection. One can think about moving this code directly into the firmware of the motor controller board.

`WheelController` compares regularly the timestamp of the last speed or power message with the current time. If the time difference is higher than a predefined security interval, the engines are stopped. The UML activity diagram is shown Figure 28.

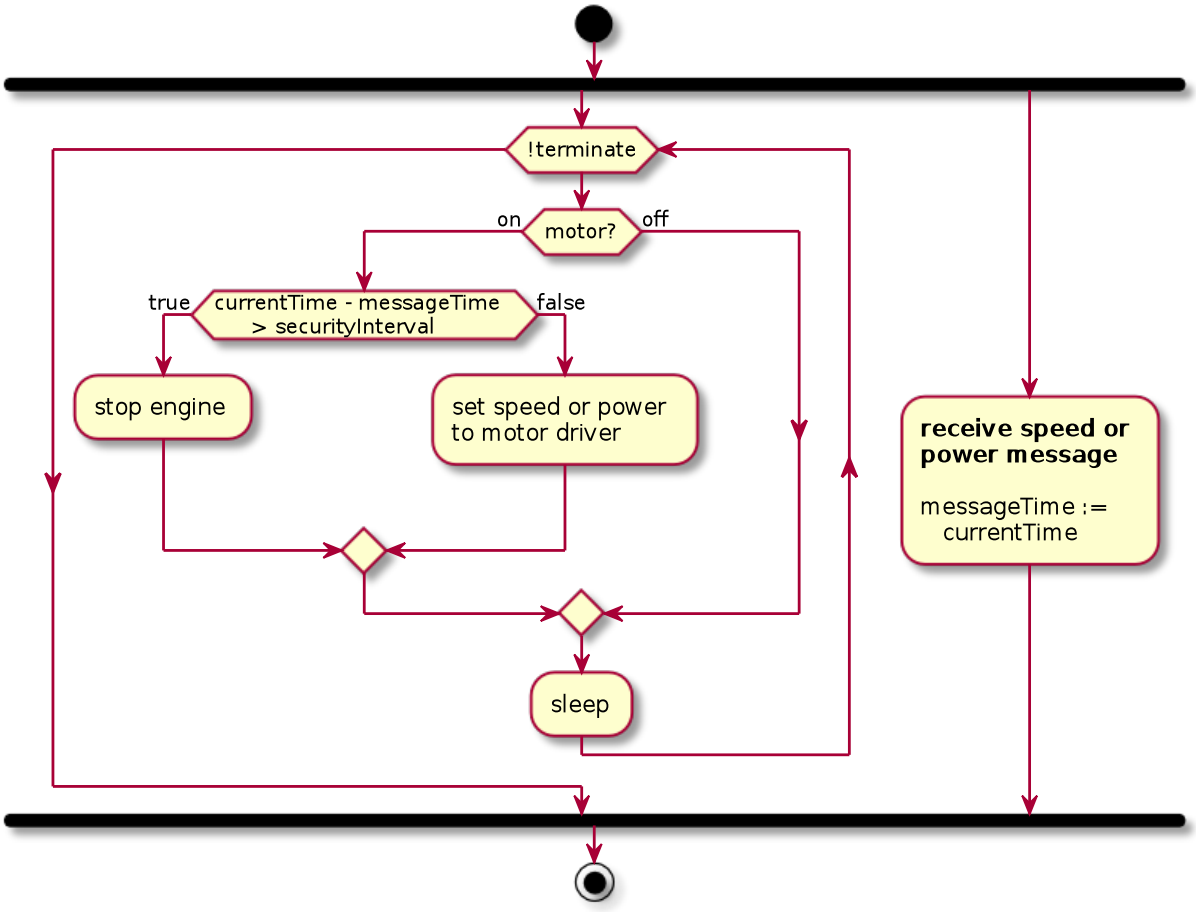


Figure 28: Wheel controller safety authority

5.2 PID Controller

The class `PController` implements a basic PID controller. The input is the target location, to which the SmartWalker must drive.

Figure 29 is the UML activity diagram for the PID control loop. The loop reads the current position from the class `Position` and computes the distance to the target location. If the walker is not at the target, the left and right speed for the wheels are computed. The speed values are then applied to the `WheelController` instance as speed metric.

Listing 2 shows the computation of the `speedLeft` and `speedRight` values from the current `position` and the target coordinates `targetX` and `targetY`.

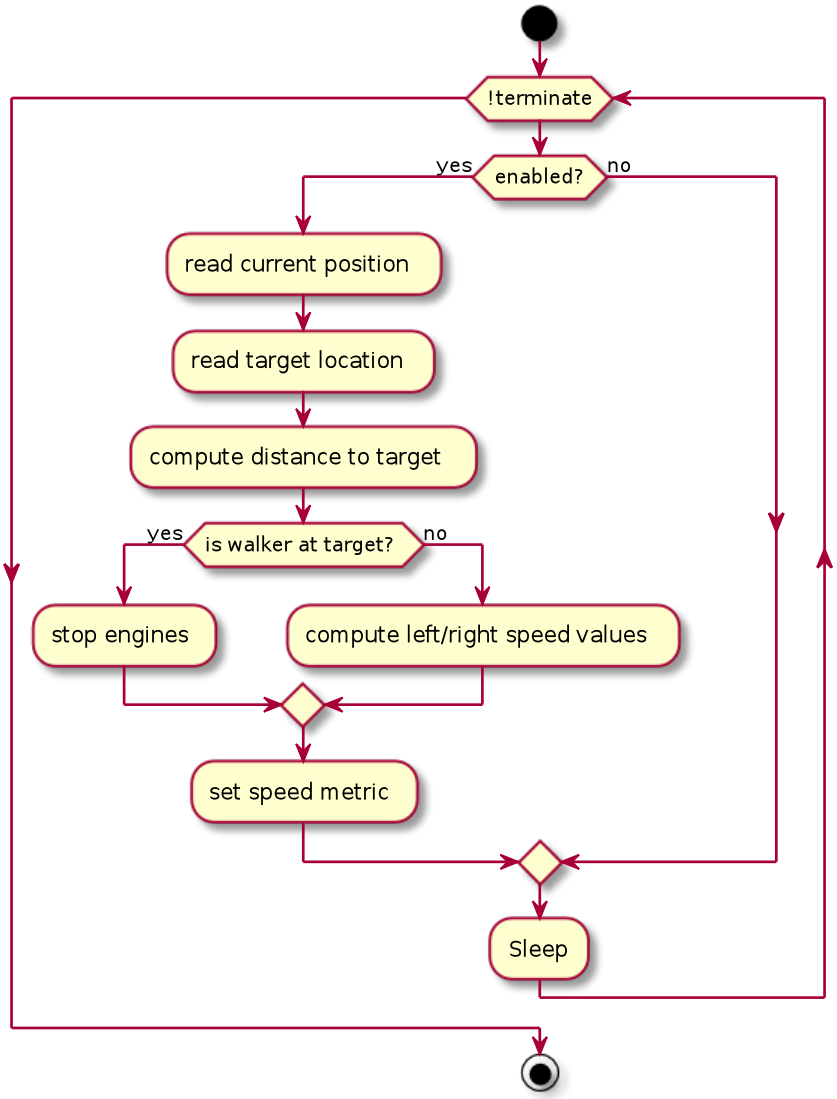


Figure 29: PID control loop

```
const double speedNom = 2.0; // nominal speed

double speedLeft;  // to compute
double speedRight; // to compute

double deltaX = targetX - position->getX();
double deltaY = targetY - position->getY();
```

```
double distance = sqrt(deltaX * deltaX + deltaY * deltaY);
double targetPhi = atan2(deltaY, deltaX);
double deltaPhi = targetPhi - position->getPhi();

if (distance < S_TOLERANCE) {
  speedLeft = 0;
  speedRight = 0;
} else {
  if (abs(deltaPhi) < M_PI/4) {
    speedLeft = speedNom * (1.0 - deltaPhi * kp);
    speedRight = speedNom * (1.0 + deltaPhi * kp);
  } else { // Turn around
    if (deltaPhi > 0) {
      speedLeft = -speedNom/4;
      speedRight = speedNom/4;
    } else {
      speedLeft = speedNom/4;
      speedRight = -speedNom/4;
    }
  }
}

if (distance < 1) { // slow down process at the end
  distance = sqrt(distance);
  speedLeft = speedLeft * distance;
  speedRight = speedRight * distance;
}

leftWheelController->setSpeedMetric(speedLeft);
rightWheelController->setSpeedMetric(speedRight);
```

Listing 2: PID speed computation

5.3 STP Controller

The speed to power (STP) control loop is the core of the assist mode. STP reduces the rolling resistance by adjust the engines speed transparently to the speed of the rollator. Depending on the inclination of the road, the distance of the user, and the brakes, this support has to be larger or smaller. Ideally, the motors would also help braking on downhill, but exhaust brake is currently not supported by the motor controller firmware.

Figure 30 shows all steps of the STP control loop. In the first step, STP reads data from the following four sensors in parallel:

- The speed values are queried from the `Speed` instances of both wheels.
- The current distance between the walker and the person, including the mean and variance, are derived from the `LegDetection` class.
- Both brakes are sending their state.
- The actual slope angle is measured by the inclinometer.

The leg detection delivers the current distance between the walker an the user. If this distance is zero, there is no one behind the walker. In that case and if one of the sensors did not deliver any data, the engines have to be stopped. Else, the resulting power for the left and the right wheel is computed as a linear combination of the values received from the sensors. These two power values are then set as power metric to the `WheelController`.

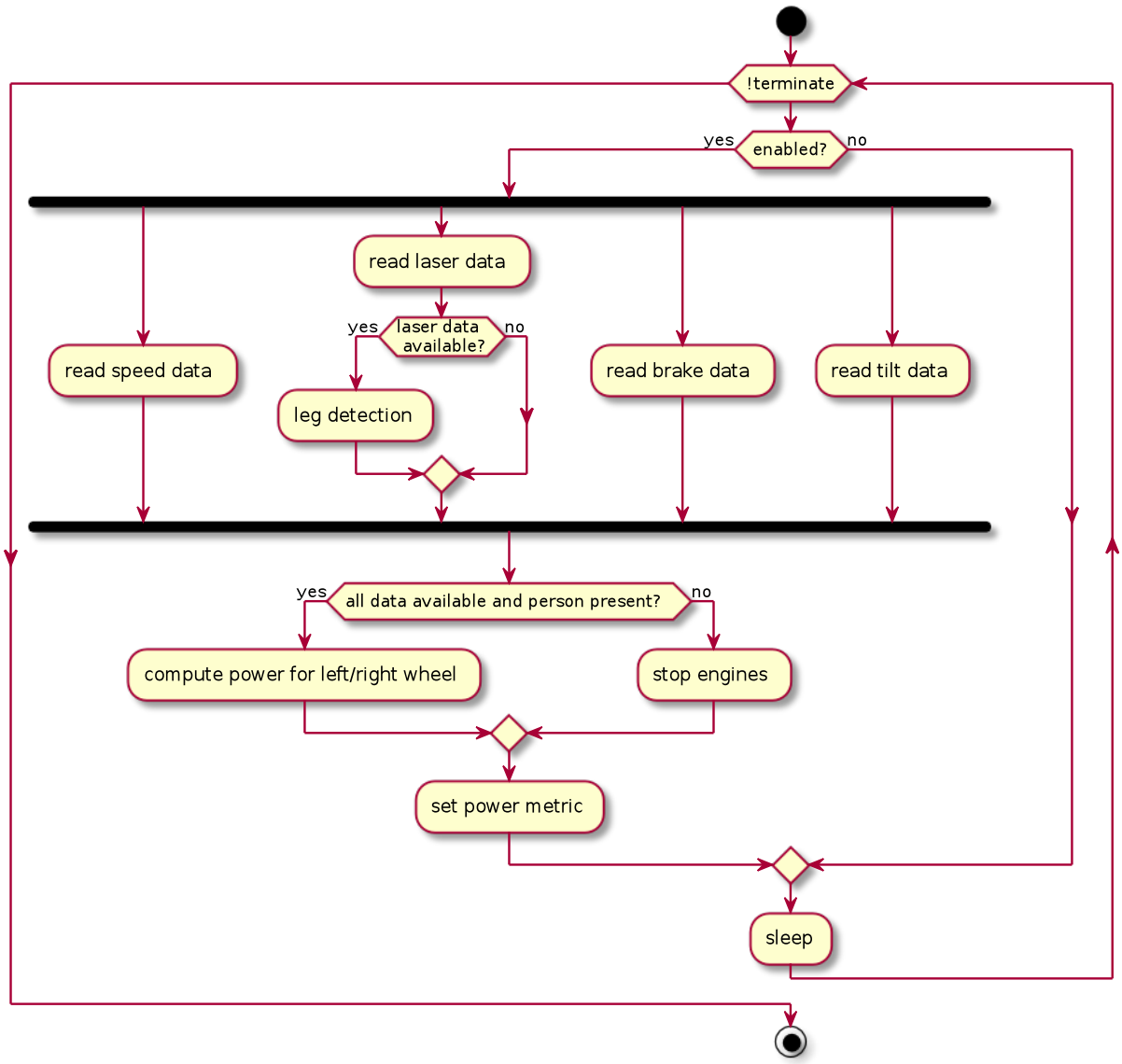


Figure 30: STP control loop

Computation of output power The following equations explain the composition of the linear combination that defines the output power per wheel.

1. Compute walker speed v from speed of left v_l and right v_r wheel. v_{fwd} is the forward speed and is set to zero when the rollator is reversing.

$$v = 0.5 \cdot (v_l + v_r)$$
$$v_{fwd} = \begin{cases} v & \text{if } v > 0 \\ 0 & \text{otherwise} \end{cases}$$

2. Compute the speed power fractions ps_l and ps_r from the speed coefficient k_v and the speed of the wheels v_l and v_r.

$$p_{s_l} = k_v \cdot v_l$$
$$p_{s_r} = k_v \cdot v_r$$

3. Compute the climb power fraction p_c, depending on speed v, pitch angle α_{pitch} and the two factors k_{ascend} and $k_{descend}$. Factor $k_{descend}$ is selected if the person walks reward downhill ($s < 0 \wedge c > 0$) or forward downhill ($s > 0 \wedge c < 0$), k_{ascend} otherwise.

$$c = \sin(\alpha_{pitch})$$
$$p_c = |v| \cdot c \cdot \begin{cases} k_{descend} & \text{if } s > 0 \oplus c > 0 \\ k_{ascend} & \text{otherwise} \end{cases}$$

4. The brake power fractions p_{b_l} and p_{b_r} are computed from the brake coefficient k_b, the brake states b_l and b_l and the negative speed $-v_l$, respectively $-v_r$. The negative speed is taken, because the brakes act in the opposite direction.

$$p_{b_l} = k_b \cdot b_l \cdot -v_l$$
$$p_{b_r} = k_b \cdot b_r \cdot -v_r$$

5. The distance power fraction p_d is only applied when the person is walking forward ($v > 0$). It depends on speed v and the distance factor k_d, the actual distance d, the mean distance $\widetilde{d}$, and also the pitch angle α_{pitch}. γ is the distance pitch threshold and is explained in paragraph **Distance Equation**.

$$\Delta d = \widetilde{d} - d$$
$$\mu = \begin{cases} -1 & \text{if } \alpha_{pitch} > \gamma \\ 0 & \text{if } \alpha_{pitch} < -\gamma \wedge \Delta d \geq 0 \\ 1 & \text{otherwise} \end{cases}$$
$$p_d = k_d \cdot v_{fwd} \cdot \Delta d \cdot \mu$$

6. Combine the four power fractions to the resulting power coefficient p_l for left and p_r for right wheel. Use the balance coefficient $k_{balance}$ to correct inequalities in engines.

$$p_l = (1 - c_{balance}) \cdot (p_{s_l} + p_c + p_{b_l} + p_d)$$
$$p_r = (c_{balance} + 1) \cdot (p_{s_r} + p_c + p_{b_r} + p_d)$$

7. Prevent the engines to turn on below some power level $c_{threshold}$, so that the motors do not start with slow movements.

$$p_l = \begin{cases} 0 & \text{if } |p_l| < c_{threshold} \\ p_l & \text{otherwise} \end{cases}$$
$$p_r = \begin{cases} 0 & \text{if } |p_r| < c_{threshold} \\ p_r & \text{otherwise} \end{cases}$$

8. Do not set the new power values to the wheel controller if there is just a minor change below c_{delta}. This prevents the motor controller from overloading if there are many small changes. The values p'_l and p'_r are the power values computed in the previous step and represent the actual value that is set to the motor controller.

$$p'_l = \begin{cases} p_l & \text{if } (p_l = 0 \wedge p'_l \neq 0) \vee sgn(p_l) \neq sgn(p'_l) \vee |p_l - p'_l| > c_{delta} \\ p'_l & \text{otherwise} \end{cases}$$
$$p'_r = \begin{cases} p_r & \text{if } (p_r = 0 \wedge p'_r \neq 0) \vee sgn(p_r) \neq sgn(p'_r) \vee |p_r - p'_r| > c_{delta} \\ p'_r & \text{otherwise} \end{cases}$$

Transition and Calibration Coefficients The five transition coefficients k_v, k_{ascend}, $k_{descend}$, k_b and k_d, and three calibration coefficients $c_{balance}$, $c_{threshold}$, and c_{delta}, are adjustable over the property system and therefore, over the virtual terminal. The values are determined heuristically during the development and are set to the following values by default:

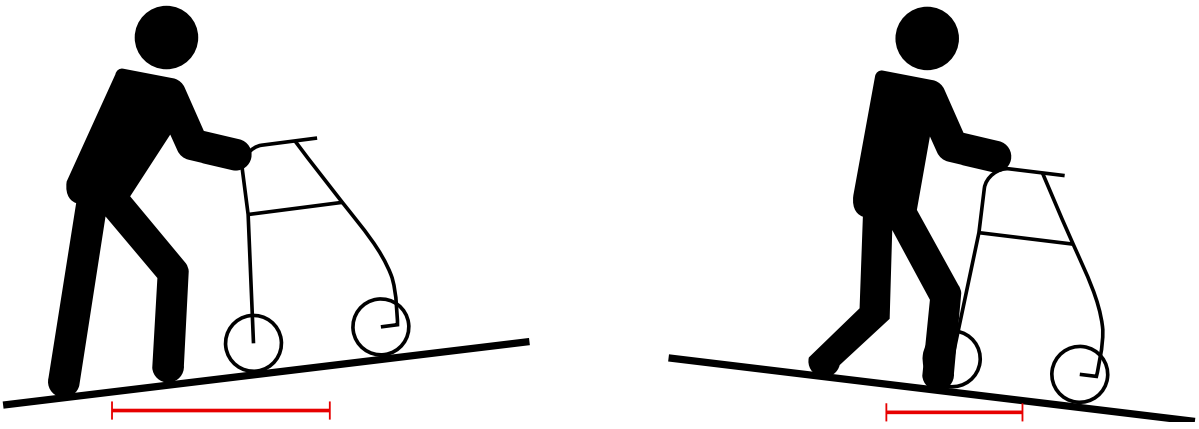
- $k_v = 6.1$
- $k_{ascend} = 114.0$
- $k_{descend} = 40.0$
- $k_b = 6.6$
- $k_d = 5.1$
- $c_{balance} = -0.1$
- $c_{threshold} = 0.05$
- $c_{delta} = 0.01$
- $\gamma = 3°$

Distance Equation Δd is the difference between the mean distance $\widetilde{d}$ and the actual distance d. Section 3.3 explained that the mean distance $\widetilde{d}$ is the mean value of all computed distances d over 90 seconds. If Δd is greater than 0, then the person is closer to the walker than usual. If Δd is less than 0, the person is walking farther away. On flat terrain, with $\Delta d > 0$, the rollator becomes slightly faster, and with $\Delta d < 0$ slower. This is because the resulting p_d is linearly dependent on Δd.

In equation 5, there is the distance pitch threshold γ, which is currently set to 3°. We determined this value heuristically during intermediate tests, but that value has to be proved in future work. The terrain is almost never perfectly flat, so the threshold γ defines the band that is considered as flat.

With the threshold γ we compute the value of μ which is either -1, 0 or 1. The definition of μ explained in short:

- $\boldsymbol{\alpha_{pitch} > \gamma \Rightarrow \mu = -1}$
  Negate the power if one moves uphill by more than γ degrees.
  On uphill, the distance to the walker is higher than on flat terrain (Figure 31a). So Δd becomes negative and the resulting p_d is negative also, therefore the walker's support would be smaller. Negating p_d now brings more power to the wheels.
- $\boldsymbol{\alpha_{pitch} < -\gamma \wedge \Delta d \geq 0 \Rightarrow \mu = 0}$
  The slope is less than $-\gamma$ degrees. On downhill, the distance between the person and the walker is shorter (Figure 31b). Δd grows and thus also the power p_d. We set p_d to zero on downhill because additional power to the engines would be dangerous.
- **otherwise** $\boldsymbol{\Rightarrow \mu = 1}$, which means flat terrain.



(a) Walking uphill

(b) Walking downhill

Figure 31: Distance variation when walking uphill or downhill

6 Evaluation

In this section we present the evaluation setup, its implementation, and the results. The goal of the evaluation was to find out if older people accept these kind of devices, what they think about the new assist mode, and which suggestions for improvement they have.

6.1 Setup

For the evaluation we created a questionnaire (Appendix B), which is divided into three parts. In the first step, the participants have to answer the first part of the questionnaire. It collects background informations as their age, the usage of a walking aid, how often someone goes outside and how often a computer, a smartphone or a similar device is used.

After answering the first part of questions, the participant walks some distance with our walker. The test takes place indoors and we do not define a special course because not all people are in equal shape. From the pilot study we learned, that some liked to walk a longer distance and others could shoot a short round only [18]. Due to the cold weather outside, it is not possible to perform some outdoor tests with the old people, why we have to postpone the inclinometer tests on increasing terrain to summer. The SmartWalker is turned off completely for the first walk, therefore the participants will feel the full weight and so the resistance of the walking aid.

Following on, the test person answers the second part of the questions about using the walker without any engine support. The questions addresses the comfort, size and weight of the SmartWalker, and the required effort to to push it (Appendix B - Part 2).

In the third part, the experimentee goes on a second round with the SmartWalker, now with the assist mode turned on. After that we ask the third set of the questions. These questions address the comfort and effort again. Their opinion on the support speed is polled, and we collect their personal preference between assistance on and off (Appendix B - Part 3).

At the end of the tests, we host an informal discussion with the participants to hear their opinion, what improvements they wish, and what they currently miss. The aim is to find out what these people generally think of such projects and devices.

6.2 Method

We tested the procedure first internally in our department. After that, we visited three nursing homes at three days, where 13 people participated in total. These number sounds not much, but one has to consider that the whole procedure is time demanding and people may get tired quite fast.

During all tests, data from the control loop (Section 5.3) is recorded to a file, which can be used for evaluation later. There are two recordings per participant, one from the walk without assistance, and a second one with. The controller recorded around 20 data rows per second, of which each row contained the following information:

- Timestamp
- Speed power fraction, one for each wheel
- Climb power fraction
- Distance power fraction
- Brake power fraction of both wheels
- Resulting power of each Wheel
- Mean distance

- Current distance
- Pitch angle
- Speed

6.3 Results & Discussion

The result section is structured into seven parts. Section 6.3.1 presents the collected background informations about the participants. This is followed by the evaluation of the walker as walking aid (Section 6.3.2). Section 6.3.3 evaluates the assist mode and Section 6.3.4 compares it with the off mode. The distance between the walker and the person is evaluated in Section 6.3.5, which is followed with the evaluation of the STP controller (Section 6.3.6). Finally, in Section 6.3.7 we summarize the feedback of the participants.

6.3.1 Participants

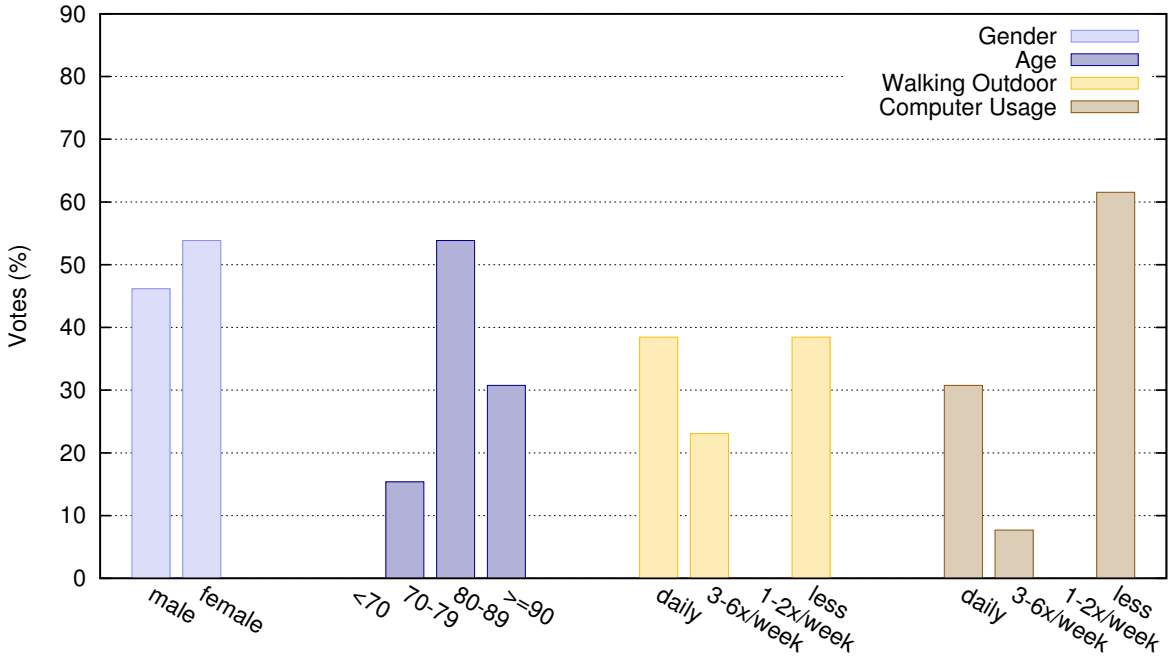


Figure 32: Informations about the participants of the evaluation

The first graph (Figure 32) shows background information about the participants. The distribution between men and women is about the same. Most people in the nursing home were aged between 80 and 89. Around 60% are walking outside daily or 15% almost daily, the remainder of 40% is less than once per week outside for a walk.

There was a clear difference between smartphone and computer users and people that never use such devices. If someone owns a smart device, it uses it daily or almost daily (40%). On the other hand, people who do not have a smart device never use one (60%). From visually impaired participants we have learned, that smart devices as reading devices are very common.

6.3.2 SmartWalker as Walking Aid

Most users refer to the walker as (too) heavy, (too) big and therefore unwieldy (Figure 33) and not really suitable for everyday life. If we compare our walker with the personal walkers of the participants, we have to agree with them. Our walker is wider, up to two times longer, and four times heavier. Section 2.1 explains why we have chosen this frame. The optimization for weight and size was not the first priority.

We have also received positive feedback concerning the weight: Some participants stated that the walker's heaviness gives some additional security and stability, compared to their current light aluminium walkers. So the heavy weight have also its advantage.

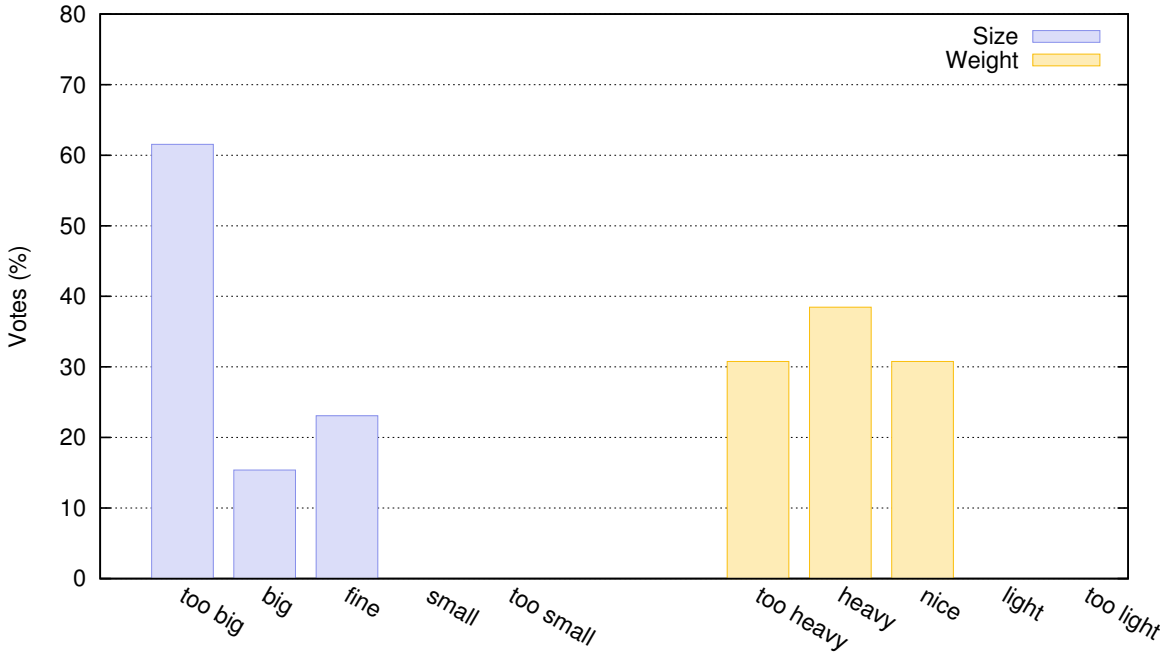


Figure 33: Evaluation of size and weight

6.3.3 Assist Mode

The speed of the assist mode is well accepted by over 80% (Figure 34). We tried to find the balance between a strong and a weak support speed, what apparently succeeded. Some people complained about the fact that the walker sometimes makes small stuttering. Our internal test revealed the same, and the likely cause is: The two hub engines (Section 2.1.2) have a minimal rotation speed. If somebody walks at a speed similar to this minimal rotation speed, the engines begin to turn off and on, which is perceptible.

Nearly 54% preferred the assist mode. On the other hand, 30% do not like the support by a robot and 16% were uncertain. Reasons for the acceptance of the assist mode are the higher walking quality and speed and the lower resistance. Next Section discusses the walking quality, the speed, and the resistance in detail.

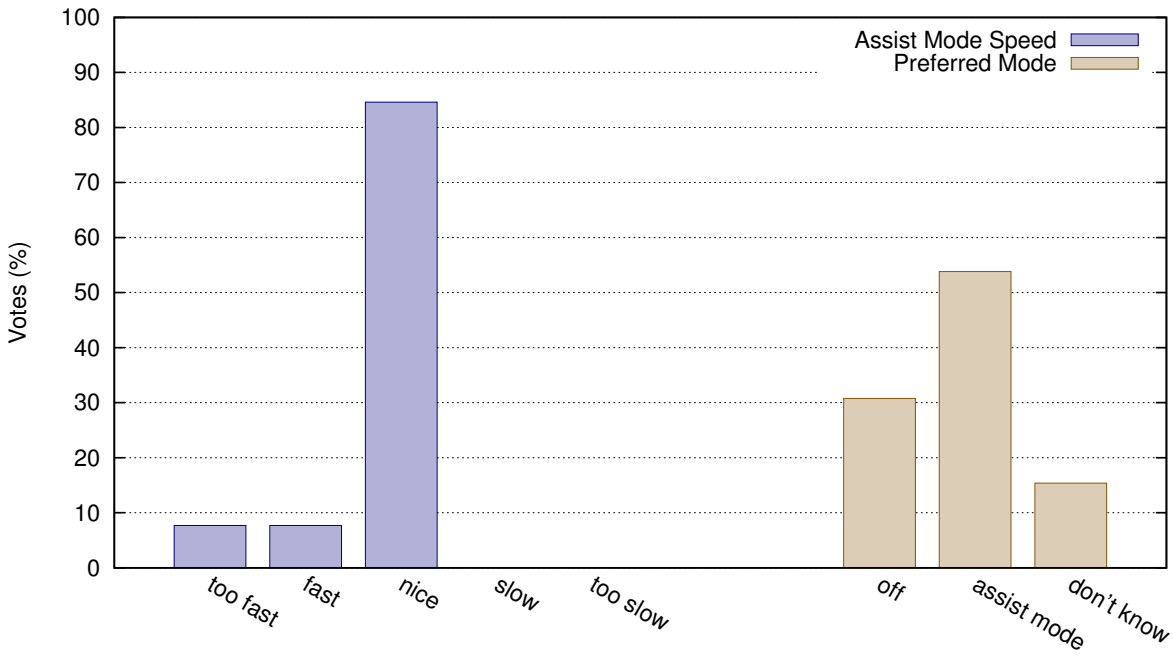


Figure 34: Evaluation of assist speed and preferred mode

6.3.4 Mode Comparison

In this section we compare the assist mode with the off mode. In all graphs, the colour of the assist mode is green, otherwise it is red.

Quality of Walking Figure 35 summarizes the answers to the question about the quality of walking. The question was asked twice, once with the walker disabled (red) and once with the assist mode on (green). The possible answers were *very uncomfortable*, *uncomfortable*, *very comfortable* and *no answer*.

Almost 70% said that walking in off mode is *comfortable*. This is quite surprising, since the majority of the participants complained about the walker's size and weight (Figure 33). We expected that walking with a heavy walker is uncomfortable.

For assist mode, more people chose the option *very comfortable*. On the other hand, also less people think it is *comfortable* now. The number of participants voting for *uncomfortable* is the same. The trend goes in the direction that the people feel the assist mode as more comfortable.

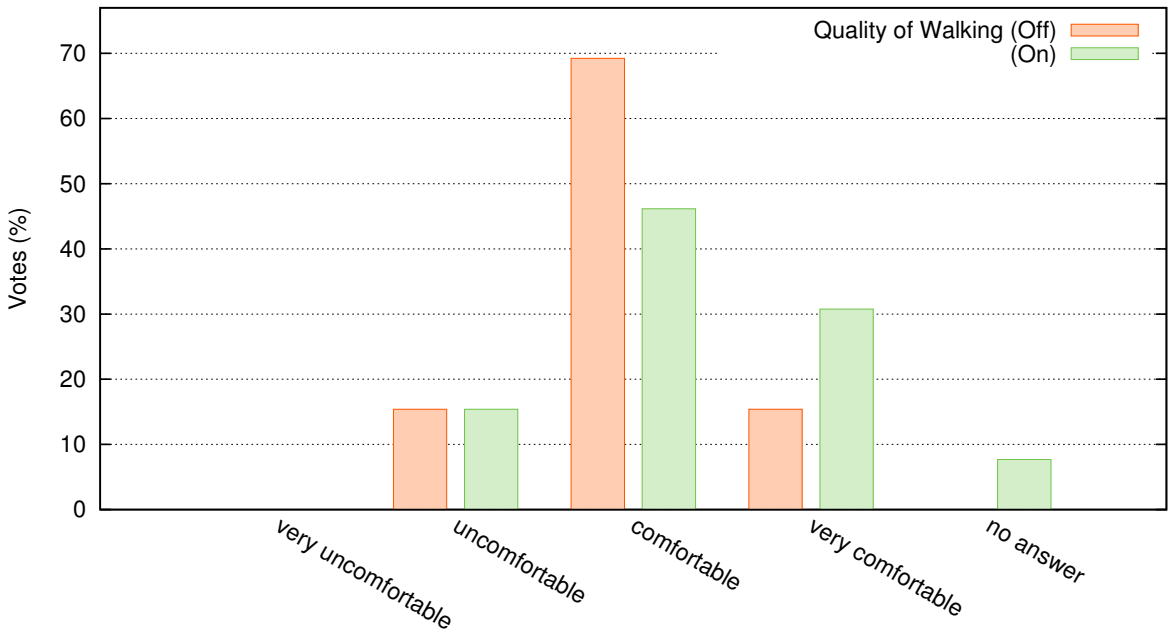


Figure 35: Evaluation of quality of walking

Resistance Figure 36 shows the opinions concerning the force required to push the *Smart-Walker* for both modes. The possible answers were *too large*, *large*, *balanced*, *small* and *too small*.

One can interpret from the answers that the assist mode requires less effort to move the walker, as expected. One person said that she felt unstable with assist mode. He supports himself heavily on the walker and therefore, he experienced less stability.

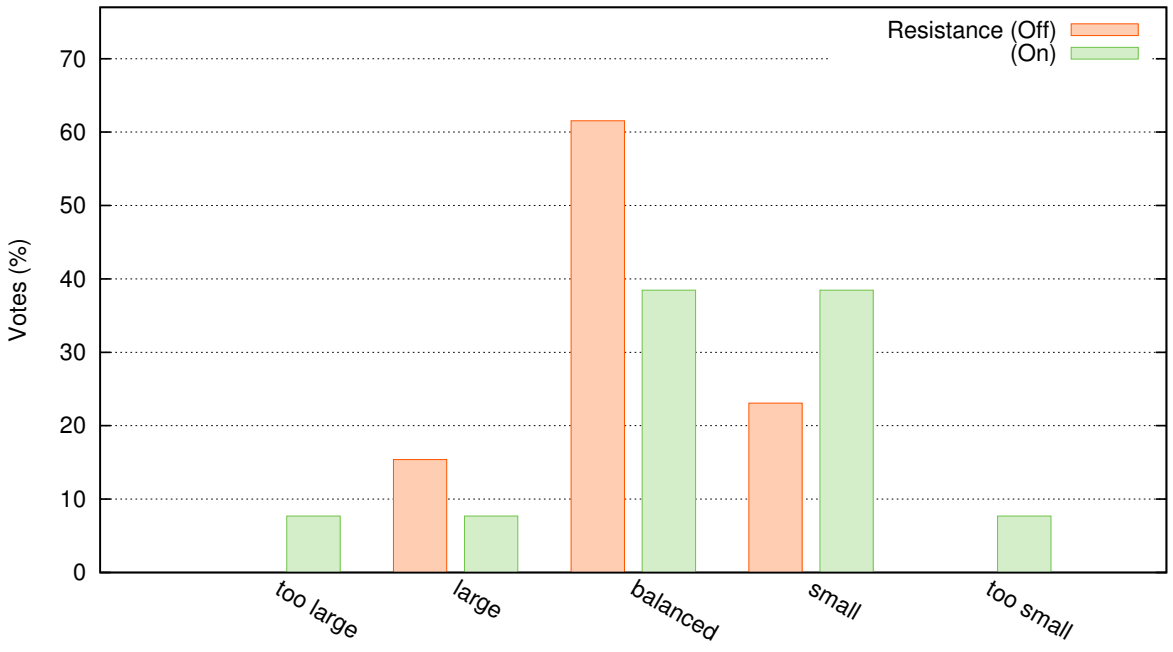


Figure 36: Evaluation of resistance

Speed In graph of Figure 37, one can see the speed fractions of both modes. Because some participants regularly paused the walking during their test, the standstill time is quite high. Apart from that, people walk slightly faster when using the assist mode. But this does not necessarily mean that assistance made them suddenly walk faster. Possibly their walking speed was slower in the first part because of the heavy weight of SmartWalker. In order to better understand the cause, one could equip their personal walker with a speedometer and compare their speed when they walk with the SmartWalker in assist mode.

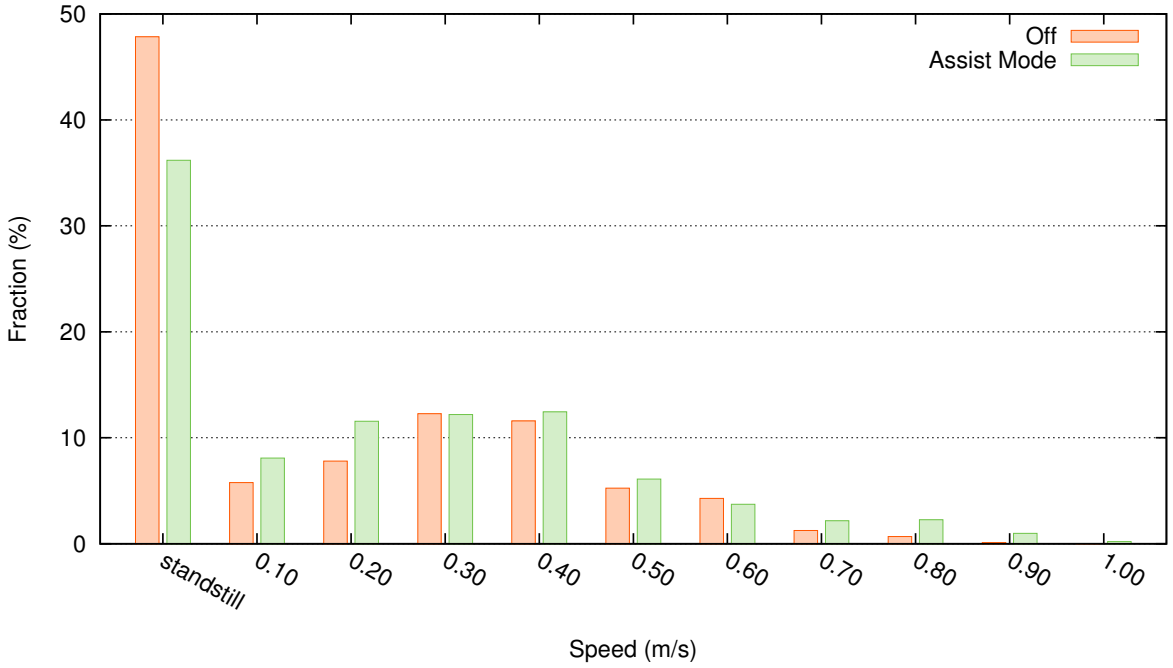


Figure 37: Walking speed distribution

6.3.5 Distance

Figure 38 shows the recorded absolute distances between the walker and the persons during their test walk. Figure 39 shows the same for for the variation. By looking at the variance, in 50% of the time it is near 0. This means that the subjects position behind the walker have remained stable for most of the time. The distribution of the absolute distance is almost flat between $40cm$ and $60cm$: People have their preferred stance and this is kept when walking.

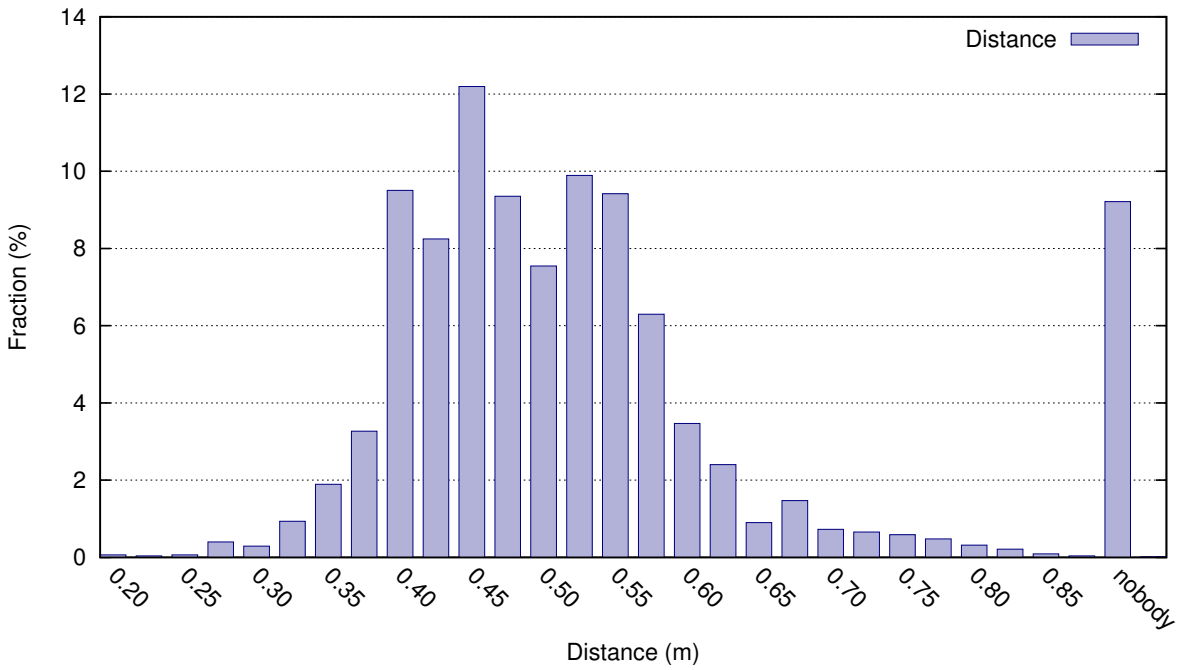


Figure 38: Distance distribution

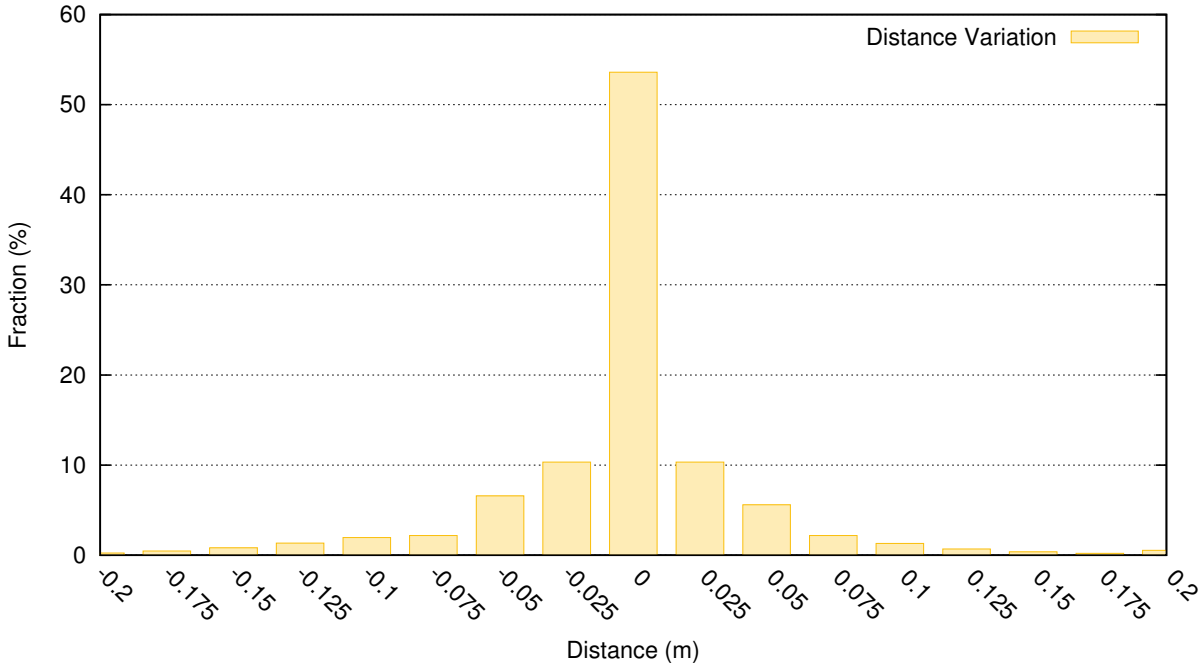


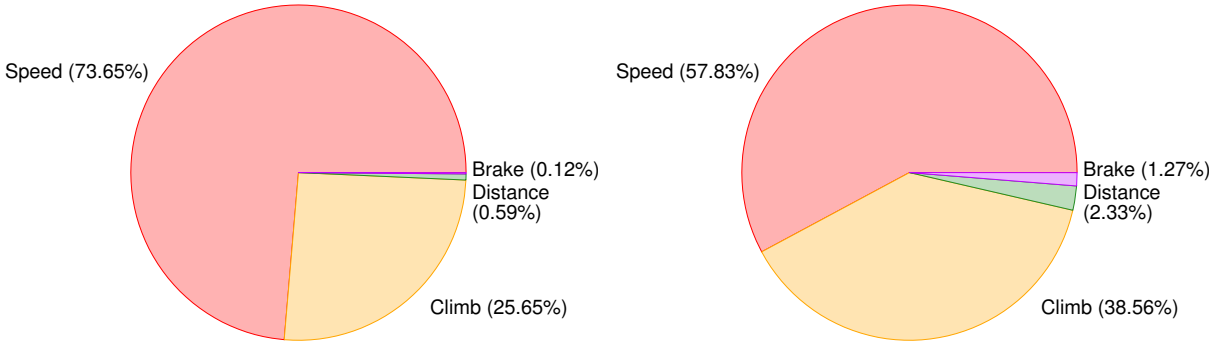
Figure 39: Distance variance distribution

6.3.6 Control

The pie chart of Figure 40 shows the magnitude of the influence of the four sensors on the final power with which the motors are operated. Section 5.3 described how the four sensors are used by the STP controller to compute the final power for the wheels. Each slice of the chart is related to the influence of one sensor.

Flat Terrain Taking a look at Figure 40a, 74% of the power is determined by the speed of the walker. Although the terrain on the elderly homes was flat, the inclinometer affected the result by 26%. It could be because the pitch sensor is vibrated when you start walking. This vibration affects the result all the time. This is not noticeable but is cumulated over time and therefore heavily weighted in the graph. On the flat terrain, almost nobody pulled the brakes, therefore this value is almost 0%. As we have seen in last Section 6.3.5, people keep their distance to the walker very stable, therefore the influence of the distance in the final power is also below 1%.

Climb The only recordings on sloping terrain are from the internal test. The climb slice of chart 40b is therefore larger. People also pulled the brake when moving downhill, so this influence is higher also.



(a) Flat terrain in elderly homes

(b) Climb of 3% slope during internal test

Figure 40: Decomposition of sensor influences

6.3.7 Feedback

Most of the participants were very interested in our project, and some even wanted to know the approximate price of the device and when we go into production. We also received very useful feedback. Most people complained about its weight, manoeuvrability, and also width. In particular, they said that its width is not very suitable for small elevators and doors.

Unfortunately, because of the cold weather, it was not possible to test outside. It would be interesting to evaluate the inclinometer with the elderly people.

7 Conclusion

This thesis presented a non-autonomous assist mode for smart walkers with speed to power (STP) as controller. Data from speed, brake and tilt sensors is used to adjust the support power of the SmartWalker. Leg detection and tracking refines the power according to the distance between the user and the walker to gain a more natural felling. There is no manual control required and there is also no additional user control equipment needed.

To accomplish this, we extended the initial version of SmartWalker with two additional sensors, an inclinometer to measure the gradient of the ground, and two brake levers with included binary pull sensors. We developed the STP controller, which is configurable by nine coefficients.

The effectiveness of the proposed control is evaluated in nursing homes. Elderly people using traditional walkers were invited to participate in a set of tests. With the help of a questionnaire and two walking tests, we analysed the acceptance, strengths and weaknesses of the SmartWalker. From these results we can confirm that our work is going in the right direction. People accepted the new assist mode, and we have received ideas and tasks for further work (Section 7.1).

We have examined the suitability of the LiDAR for environment tracking (Section 4). From these tests we learned, that it can not work, because the CSM algorithm requires a much more accurate input. The resolution of the XV-11 LiDAR is to small at distances beyond $2m$ and a big part of the view range is covered by the wheels of the SmartWalker.

7.1 Future Work

There are many possibilities to improve the SmartWalker. Concerning the assist mode, the highest priority should be assigned to the motors and the motor controller. It has to be evaluated whether the motor controller firmware can be adapted to support braking and slower speeds.

The second priority is the implementation of collision avoidance, not only for the autonomous mode, but also for the non-autonomous assist mode. We noticed, that people often touched door frames, chairs and other obstacles with our walker. Maybe we can help people to avoid that with collision detection.

The third priority is the replacement of the inclinometer. With a two-axis inclinometer (Section 2.1.5) it would be possible to measure the lateral inclination. With this information we are able to compensate the drift to the left or right side when walking on a longitudinal slope.

More Ideas for future work are listed below with a few keywords.

- Software features
  - **User interface** Easy to use interface, maybe with integrated alarm button, phone and map routing.
  - **Collision detection** Use the LiDAR for collision detection and avoidance
  - **Service mode** The SmartWalker follows his owner (See Kim, Chung, and Yoo [20])
- Sensors
  - **Tilt sensor** Two-axis inclinometer (Section 2.1.5).
  - **Handlebar touch sensor** With a handlebar touch sensor we could detect if someone releases the walker.
  - **GPS** Dedicated GPS receiver for outdoor positioning.

  - **Weight sensor** The STP controller is not aware about the weight of the SmartWalker. If this weight changes, then the coefficients have to be adjusted. With a weight sensor this can be done automatically.

- Evaluation
  - **Outdoor tests** Perform outdoor tests for testing the inclinometer
  - **Personal walker** Compare the SmartWalker's assist mode with the personal walker. Right now, the assist mode was compared only with the disabled walker.

- Mechanics
  - **Weight** Lighter frame
  - **Size** Smaller wheelbase
  - **Handling** Foldable frame

7.2 What I have learned

I liked to work in an interdisciplinary way, not just in terms of technology, computer science and mechanical and electrical engineering, but also in terms of social contacts.

This project promised to be exciting from the start, and so it was. The largest part affected software engineering, which is my main area, but it also required some knowledge in mechanics and electronics. Although good skills in mechanics and electronics were not directly a prerequisite for the success, in the end it was beneficial to understand the hardware in detail. To be honest, it was the only way to effectively develop a software solution for a very hardware-level product and to attach proposals for extensions.

For a software engineer, the SmartWalker has an unusual target group, namely people with mobility problems and therefore mostly old people, which are dependent on a walker. This required to listen to these people, explain them our ideas and pick up their mind about. Such non commonplace social contacts were challenge and opportunity for the same.

The initial idea of controlling the walker's speed just over leg detection had to be refined quickly. In practice, when using leg tracking only, latencies are too high because the speed of the LiDAR is low and there are many uncertainties in detecting and tracking legs. When I started with the implementation of the control algorithms (Section 5), the speed of the SmartWalker was mainly determined over the leg detection algorithm (Section 3). I reduced this influence step by step and finally I ended up with the STP (Speed To Power) algorithm, described in Section 5.3.

This experience showed me, that you can not stick to the first ideas.

Appendices

A Implemented EM Algorithm

```
const static double LDA_DIMENSION = 3;
const static double LDA_FACTOR = pow(2*M_PI, 0.5*LDA_DIMENSION);

/**
 * EM Alogrithm for 2 regions
 *
 * @param cloud array of points
 * @param max_iterations maximum iterations to use
 * @param delta if the change from one iteration step to the next is less than delta ->↩
     stop
 * @param dim1 search dimension for center 1
 * @param dim2 search dimension for center 2
 * @param center1 (in/out) initial position of center 1
 * @param center1 (in/out) initial position of center 2
 * @return number of iterations
 */
int emlda(const vector<Vector2> &cloud, const int max_iterations, const double delta,
        const Vector2 &dim1, const Vector2 &dim2, Vector2 &center1, Vector2 &center2) {

    if (cloud.empty() || max_iterations < 1 || delta < 0)
        return 0;

    // initialize covariance matrices
    Matrix2 cov1; cov1 = dim1;
    Matrix2 cov2; cov2 = dim2;

    // initialize probability matrix
    const int num_points = cloud.size();
    vector<double> probs1(num_points);
    vector<double> probs2(num_points);

    int iteration = 0;
    while (iteration++ < max_iterations) {

        double det1, det2;
        Matrix2 icov1 = cov1.inverse_and_determinant(det1);
        Matrix2 icov2 = cov2.inverse_and_determinant(det2);

        const double pa1 = 1.0/(LDA_FACTOR*sqrt(det1));
        const double pa2 = 1.0/(LDA_FACTOR*sqrt(det2));

        for (int pid=0; pid<num_points; pid++) {
            const Vector2 p = cloud[pid];

            const Vector2 d1 = p - center1;
            const double pb1 = exp(-0.5*d1*(icov1*d1));
            probs1[pid] = pa1*pb1;

            const Vector2 d2 = p - center2;
            const double pb2 = exp(-0.5*d2*(icov2*d2));
            probs2[pid] = pa2*pb2;
        }

        // normalize probabilities
        for (int pid=0; pid<num_points; pid++) {
            double sum_p = probs1[pid] + probs2[pid];
            probs1[pid] /= sum_p;
            probs2[pid] /= sum_p;
        }

        // compute mean and covariance statistics for each class
        double mean_change = 0;
        double sum_p1 = 0;
        double sum_p2 = 0;
```

```
        // compute mean vector
        Vector2 mean1, mean2;
        for (int pid=0; pid<num_points; pid++) {
            const Vector2 p = cloud[pid];

            const double p1 = probs1[pid];
            mean1 += p1*p;
            sum_p1 += p1;

            const double p2 = probs2[pid];
            mean2 += p2*p;
            sum_p2 += p2;
        }
        mean1 /= sum_p1;
        mean2 /= sum_p2;

        // compute covariance matrix (pooled covariance estimate)
        cov1 = 0;
        cov2 = 0;
        for (int pid=0; pid<num_points; pid++) {
            const Vector2 p = cloud[pid];

            const double p1 = probs1[pid];
            const Vector2 d1 = p - mean1;
            cov1 += d1^(p1*d1);

            const double p2 = probs2[pid];
            const Vector2 d2 = p - mean2;
            cov2 += d2^(p2*d2);
        }
        cov1 /= sum_p1;
        cov2 /= sum_p2;

        mean_change +=
            0.25*(mean1 - center1).abs().sum() +
            0.25*(mean2 - center2).abs().sum();

        center1 = mean1;
        center2 = mean2;

        // test if the mean change is below a certian delta
        if (mean_change <= delta)
            break;
    }

    return iteration;
}
```

Listing 3: EM algorithm implementation

B Original Questionnaire

Teil 1 – Fragen zur Person

1. ❍ Männlich ❍ Weiblich
2. Wie alt sind Sie?
   ❍ unter 70 ❍ 70 – 79 ❍ 80 – 89 ❍ über 90
3. Benutzen Sie eine Gehhilfe?
   ❍ Nein ❍ Stock ❍ Rollator ❍ Rollstuhl
4. Wenn ja, wie lange benutzen sie die Gehhilfe bereits?
   ❍ <1 Jahr ❍ 1 – 2 Jahre ❍ 3 – 5 Jahre ❍ mehr als 5 Jahre
5. Wie oft benutzen Sie die Gehhilfe?
   ❍ täglich ❍ 4-6x / Woche ❍ wöchentlich ❍ weniger
6. Wie oft gehen Sie nach draussen?
   ❍ täglich ❍ 4-6x / Woche ❍ wöchentlich ❍ weniger
7. Wie oft benutzen Sie einen Computer oder ähnliche Geräte (Smartphone)?
   ❍ täglich ❍ 4-6x / Woche ❍ wöchentlich ❍ weniger

Teil 2 – Benutzung des Rollators **ohne** Motorunterstützung

8. Wie angenehm ist es mit dem Rollator zu gehen?
   ❍ sehr unang. ❍ unangenehm ❍ angenehm ❍ sehr ang. ❍ weiss nicht
9. Wie empfinden Sie die Grösse des Rollators?
   ❍ zu gross ❍ gross ❍ angenehm ❍ klein ❍ zu klein
10. Wie empfinden Sie das Gewicht?
    ❍ zu schwer ❍ schwer ❍ angenehm ❍ leicht ❍ zu leicht
11. Wie empfinden Sie den Kraftaufwand, um den Rollator zu schieben?
    ❍ zu gross ❍ gross ❍ angenehm ❍ klein ❍ zu klein

Teil 3 – Benutzung des Rollators **mit** Motorunterstützung

12. Wie angenehm ist es mit dem Rollator **mit Motorunterstützung** zu gehen?
    ❍ sehr unang. ❍ unangenehm ❍ angenehm ❍ sehr ang. ❍ weiss nicht
13. Wie empfinden sie die Geschwindigkeit des Rollators?
    ❍ zu schnell ❍ schnell ❍ angenehm ❍ langsam ❍ zu langsam
14. Wie empfinden Sie jetzt den Kraftaufwand, um den Rollator zu schieben?
    ❍ zu gross ❍ gross ❍ angenehm ❍ klein ❍ zu klein
15. Finden sie es angenehmer mit oder ohne Motorunterstützung zu gehen?
    ❍ **ohne** Unterstützung ❍ **mit** Unterstützung ❍ weiss nicht
16. Wieso empfinden sie es als **unangenehm** mit Motorunterstützung zu gehen?
    ❑ Kein Vertrauen ❑ Reaktionszeit ist zu tief ❑ _____
17. Würden Sie unseren Rollator ihrem jetzigen Rollator vorziehen?
    ❍ ja ❍ eher ja ❍ eher nein ❍ nein ❍ weiss nicht

References

[1] *BeagleBone Black*. BeagleBoard.org. URL: http://beagleboard.org/BLACK.

[2] P.J. Besl and Neil D. McKay. "A method for registration of 3-D shapes". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 14.2 (Feb. 1992), pp. 239–256. ISSN: 0162-8828. DOI: 10.1109/34.121791.

[3] *Bridging (networking)*. Wikipedia. URL: http://en.wikipedia.org/wiki/Bridging_(networking). Network bridging is the action taken by network equipment to create an aggregate network from either two or more communication networks.

[4] *Brushless DC (BLDC) Motor RDK*. Texas Instruments. URL: http://www.ti.com/tool/RDK-BLDC.

[5] Andrea Censi. "An ICP variant using a point-to-line metric". In: *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. May 2008, pp. 19–25. DOI: 10.1109/ROBOT.2008.4543181.

[6] Y. Chen and G. Medioni. "Object modeling by registration of multiple range images". In: *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*. Apr. 1991, 2724–2729 vol.3. DOI: 10.1109/ROBOT.1991.132043.

[7] Jormungand Chris. *EM Clustering Algorithm*. URL: http://jormungand.net/projects/misc/em/.

[8] *Cluster analysis*. Wikipedia. URL: http://en.wikipedia.org/wiki/Cluster_analysis. Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters).

[9] *CSM*. Massachusetts Institute of Technology. URL: http://censi.mit.edu/software/csm/. The C(anonical) Scan Matcher (CSM) is a pure C implementation of a very fast variation of ICP using a point-to-line metric optimized for range-finder scan matching.

[10] A. P. Dempster, N. M. Laird, and D. B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm". In: *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B* 39.1 (1977), pp. 1–38.

[11] *Eiffel Software*. Eiffel Software. URL: https://www.eiffel.com/.

[12] A. Elias et al. "Robotic walkers from a clinical point of view: Feature-based classification and proposal of the UFES Walker". In: *Biosignals and Biorobotics Conference (BRC), 2012 ISSNIP*. Jan. 2012, pp. 1–5. DOI: 10.1109/BRC.2012.6222155.

[13] *Expectation–maximization algorithm*. Wikipedia. URL: http://en.wikipedia.org/wiki/Expectation-maximization_algorithm. In statistics, an expectation–maximization (EM) algorithm is an iterative method for finding maximum likelihood or maximum a posteriori (MAP) estimates of parameters in statistical models.

[14] *Hacking the Neato XV-11*. Wikispaces. URL: https://xv11hacking.wikispaces.com/.

[15] *Hall effect sensor*. Wikipedia. URL: http://en.wikipedia.org/wiki/Hall_effect_sensor. A Hall effect sensor is a transducer that varies its output voltage in response to a magnetic field. Hall effect sensors are used for proximity switching, positioning, speed detection, and current sensing applications.

[16] *HTML5*. Wikipedia. URL: http://en.wikipedia.org/wiki/HTML5. HTML5 is a core technology markup language of the Internet used for structuring and presenting content for the World Wide Web.

[17] *INI File*. Wikipedia. URL: http://en.wikipedia.org/wiki/INI_file. The INI file format is an informal standard for configuration files for some platforms or software. INI files are simple text files with a basic structure composed of sections, properties, and values.

[18] David Itten. "Gesture-based user interface for SmartWalker". ETH Zürich, Chair of Software Engineering, Sept. 2014.

[19] R. E. Kalman. "A New Approach to Linear Filtering And Prediction Problems". In: *ASME Journal of Basic Engineering* (1960).

[20] Hoyeon Kim, Woojin Chung, and Yoonkyu Yoo. "Detection and tracking of human legs for a mobile service robot". In: *Advanced Intelligent Mechatronics (AIM), 2010 IEEE/ASME International Conference on*. July 2010, pp. 812–817. DOI: 10.1109/AIM.2010.5695844.

[21] Geunho Lee et al. "JAIST Robotic Walker control based on a two-layered Kalman filter". In: *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. May 2011, pp. 3682–3687. DOI: 10.1109/ICRA.2011.5979784.

[22] *Lidar*. Wikipedia. URL: http://en.wikipedia.org/wiki/Lidar. Lidar (also written LIDAR, LiDAR or LADAR) is a remote sensing technology that measures distance by illuminating a target with a laser and analyzing the reflected light.

[23] Feng Lu and E.E. Milios. "Robot pose estimation in unknown environments by matching 2D range scans". In: *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*. June 1994, pp. 935–938. DOI: 10.1109/CVPR.1994.323928.

[24] Geoffrey J. McLachlan. *Discriminant analysis and statistical pattern recognition*. Wiley series in probability and mathematical statistics. New York, Chichester, Brisbane: J. Wiley and sons, 1992. ISBN: 0-471-61531-5. URL: http://opac.inria.fr/record=b1077973.

[25] B. Meyer. "Applying 'design by contract'". In: *Computer* 25.10 (1992), pp. 40–51. ISSN: 0018-9162. DOI: 10.1109/2.161279.

[26] *Neato Robotics*. Neato Robotics. URL: http://www.neatorobotics.com.

[27] Nienaltowski and Piotr. "Practical framework for contract-based concurrent object-oriented programming". PhD thesis. ETH Zürich, 2007.

[28] *Open Web Platform Milestone Achieved with HTML5 Recommendation*. W3C. URL: http://www.w3.org/2014/10/html5-rec.html.en.

[29] *PEI-Z100-AL-232-1 360° inclinometer*. PEWATRON Online Shop. URL: http://shop.pewatron.com/search/pei-z100-al-232-1-360%C2%B0-inclinometer.htm.

[30] *PEI-Z245-AL-232-1-17, 2 axis ± 45°*. PEWATRON Online Shop. URL: http://shop.pewatron.com/search/pei-z245-al-232-1-17--2-axis.htm.

[31] *POCO C++ Libraries*. POCO. URL: http://pocoproject.org/. Modern, powerful open source C++ class libraries and frameworks for building network- and internet-based applications that run on desktop, server, mobile and embedded systems.

[32] O. Postolache et al. "Smart walker for pervasive healthcare". In: *Sensing Technology (ICST), 2011 Fifth International Conference on*. Nov. 2011, pp. 482–487. DOI: 10.1109/ICSensT.2011.6137027.

[33] *Prepackaged ARM HF Linux Images*. ARMhf. URL: http://www.armhf.com. Easy to deploy ARM HF Linux images for BeagleBone, ODROID-XU, and Wandboard devices.

[34] *PrimeSense*. Wikipedia. URL: http://en.wikipedia.org/wiki/PrimeSense.

[35] Morgan Quigley et al. “ROS: an open-source Robot Operating System”. In: *ICRA workshop on open source software*. Vol. 3. 3.2. 2009, p. 5.

[36] *Reed switch*. Wikipedia. URL: http://en.wikipedia.org/wiki/Reed_switch. The reed switch is an electrical switch operated by an applied magnetic field.

[37] *Rim brake*. Wikipedia. URL: http://en.wikipedia.org/wiki/Bicycle_brake#Rim_brakes. Rim brakes apply the force by friction pads to the rim of the rotating wheel.

[38] *Roboscoop Research Project*. Chair of Software Engineering. URL: http://se.inf.ethz.ch/research/roboscoop/. Roboscoop is a research project of the Chair of Software Engineering at ETH Zurich and iHomeLab at Hochschule Luzern; the Autonomous Systems Lab at ETH Zurich serves as a project advisor, bringing its own experience in autonomous robots. The aim of Roboscoop is to improve the tools and techniques for developing robotics software. The main demonstrator application is a smart walker robot for Ambient Assisted Living.

[39] *ROS Indigo Igloo*. ROS. URL: http://wiki.ros.org/indigo. ROS Indigo Igloo is the eighth ROS distribution release and was released July 22nd, 2014.

[40] Luca Tausel et al. “Human-walker interaction on slopes based on LRF and IMU sensors”. In: *Biomedical Robotics and Biomechatronics (2014 5th IEEE RAS EMBS International Conference on*. July 2014, pp. 227–232. DOI: 10.1109/BIOROB.2014.6913781.

[41] *Tektro EL550-RS Brake Lever*. Tektro. URL: http://www.tektro.com/_english/01_products/01_prodetail.php?pid=199&sortname=Lever&sort=1&fid=3.

[42] *Telnet*. Wikipedia. URL: http://en.wikipedia.org/wiki/Telnet. Telnet is a network protocol used on the Internet or local area networks to provide a bidirectional interactive text-oriented communication facility using a virtual terminal connection.

[43] *Trionic Veloped - the walker for active people*. Trionic. URL: http://www.trionic.se/. Veloped is the modern alternative to a rollator - offering greater access and better comfort.

[44] Robert C. Tryon. *Cluster Analysis: Correlation Profile and Orthometric (factor) Analysis for the Isolation of Unities in Mind and Personality. Edwards Brothers*. 1939.

[45] *Web server*. Wikipedia. URL: http://en.wikipedia.org/wiki/Web_server. A web server is a computer system that processes requests via HTTP, the basic network protocol used to distribute information on the World Wide Web.

[46] *WebSocket*. Wikipedia. URL: http://en.wikipedia.org/wiki/WebSocket. WebSocket is a protocol providing full-duplex communications channels over a single TCP connection. The WebSocket protocol was standardized by the IETF as RFC 6455 in 2011.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Eigenständigkeitserklärung

Die unterzeichnete Eigenständigkeitserklärung ist Bestandteil jeder während des Studiums verfassten Semester-, Bachelor- und Master-Arbeit oder anderen Abschlussarbeit (auch der jeweils elektronischen Version).

Die Dozentinnen und Dozenten können auch für andere bei ihnen verfasste schriftliche Arbeiten eine Eigenständigkeitserklärung verlangen.

Ich bestätige, die vorliegende Arbeit selbständig und in eigenen Worten verfasst zu haben. Davon ausgenommen sind sprachliche und inhaltliche Korrekturvorschläge durch die Betreuer und Betreuerinnen der Arbeit.

Titel der Arbeit (in Druckschrift):

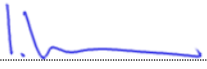
Robot control by user tracking with a laser range scanner

Verfasst von (in Druckschrift):
Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich.

Name(n):	**Vorname(n):**
Steinmann	Ivo

Ich bestätige mit meiner Unterschrift:
- Ich habe keine im Merkblatt „Zitier-Knigge“ beschriebene Form des Plagiats begangen.
- Ich habe alle Methoden, Daten und Arbeitsabläufe wahrheitsgetreu dokumentiert.
- Ich habe keine Daten manipuliert.
- Ich habe alle Personen erwähnt, welche die Arbeit wesentlich unterstützt haben.

Ich nehme zur Kenntnis, dass die Arbeit mit elektronischen Hilfsmitteln auf Plagiate überprüft werden kann.

Ort, Datum	**Unterschrift(en)**
Zürich, 27. Februar 2015	

Bei Gruppenarbeiten sind die Namen aller Verfasserinnen und Verfasser erforderlich. Durch die Unterschriften bürgen sie gemeinsam für den gesamten Inhalt dieser schriftlichen Arbeit.