**Advice document**

**Social distancing robot.**

**Fontys**

**Eindhoven**

|  |
| --- |
| **Project owner: Cees van Tilborg** |
| **Dat****e: 08-01-2020** |

|  |
| --- |
| **Project team:** |
| **Yves Lardenoije, Atanas Marchev, Bram Odenthal, Hugo Vaessen, Nicky Verhees, Martijn Vissers** |

#### Version

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Editor(s)** | **Edits** | **Status** |
| V1 | 08-01-2021 | Hugo Vaessen, Martijn Vissers, | Start document | Concept |
|  |  |  |  |  |

**Distribution**

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **To** |
|  |  |  |
|  |  |  |

**Index**

[About this document 4](#_Toc61512880)

[Simulation 4](#_Toc61512881)

[Pathfinding 4](#_Toc61512882)

[Robot behaviour 4](#_Toc61512883)

[Code 5](#_Toc61512884)

[RobotCLI 5](#_Toc61512885)

[Server 5](#_Toc61512886)

[Client 5](#_Toc61512887)

[Summary 5](#_Toc61512888)

# About this document

This is the advice document for the Rigs project, our advice is given on the current state of the project and on how the next group could possible proceed further on our work.

# Simulation

Since the hardware might not be always available, or in our case, not available at all, we decided to create a simulation environment to test the software. Currently we have made the model of the robot for the simulation. We have implemented the lidar and the movement node.

We recommend the next group to finish the simulation until the robot is completely simulated.

**Movement**

The movement node code is mostly a sketch that works as a proof of concept. It has a follow the carrot algorithm that needs to be adjusted to follow a path instead of just a goal. This means that there needs to be adjustments made in the follow\_carrot.cpp file. It is also advised to use and tune the PID controllers for a smoother movement. Currently the movement node uses the torque limitation of the robot, this needs to be fixed. It might also be a good idea to refactor the code of this node. For more information please read README\_MOVEMENT.md

# Pathfinding

Currently the pathfinding works only on a separate program because we did not have time to implement it in the simulation and test it. The algorithm can be optimized more and will have problems at the beginning because we could not test it with the robot. We would recommend trying different turning angles and choosing the ones that work the best with the robot. We would also recommend adding more input possibilities for the forward and backwards movement since the robot is going to carry spillable items.

We would also recommend creating a simple map editing tool that would allow for making irregular maps fast and easy.

# Robot behaviour

Currently, the robot has a camera installed. The main purpose for this camera is to check whether any obstacle found by the lidar is a human. This is currently not yet implemented in the software. We would recommend using the facial recognition software to know if an obstacle is a human. When a human is identified, the robot should sometimes take different actions, for example:

* Signalling the human that is has been seen (for example, with the led strip)
* Choosing another route to avoid the human
* Stopping and waiting for the human to pass

It might also be interesting if the robot could differentiate between adults and children, thus also acting in different ways.

# Code

The repository of the Rigs project contains code examples and the skeleton base for the robot. Inside is a proof of concept that needs to be extended further for future use on the robot and to communicate with the bartender and clients (which are the customers in this case).

The repository also has a configured continuous integration feature that only needs a new agent once it is given to the next group. The next group could use it to track if the current head version of the project is compiling correctly and if it passes any test that are configured by the group.

Information about the system design of the robot and its components can be found in the following documents:

* Communication Specification Document.
* Database Specification Document
* Order System Proposal

These documents are not finalized and can be used as blueprint to setup the actual environment of the Robot and the according systems. Thus, it would not be required to follow the information provided in these documents but can be useful in determining in how to proceed on these subjects.

## RobotCLI

The code for the robot is inside the repository ‘robot’ folder. This is a C++ project type. All interfaces of the hardware abstraction layer are inside the include folder of the project root. Most interfaces have been documented and can be implemented to work with the received hardware at a later stage.

The main.cpp will start the RigsController which will act as the general controller of the robot and accesses the interfaces defined in the include folder. This should be extended properly.

The RigsController has an active connection to the bartender server and needs to act on instructions received on this socket. For this event handling should be implemented to make this work smoother and instruct the robot on what operations to perform.

## Server

The server is written in Python. This project would act as server between the robot and the customers. Operations such as storing orders, sending orders to the robot, and billing will be handled from here.

Both the server and client act as proof of concept and could be replaced by any other application of choice if there is the possibility to connect to the sockets.

## Client

The client is also written in Python and can be used to send an order to the server, which in turn will send the order to the robot for further processing. This small project is also a proof of concept and is not bound to Python. Any implementation will do if the socket support is available, and the usage of the Communication Specification Document is kept in mind.

## Summary

As for the code of the rigs project, the RobotCLI should be improved and extended as of the current version. The client and server of the project are simple test implementations that can be used to further extend or can be discarded but used for reference in the new implementation of the client and server.