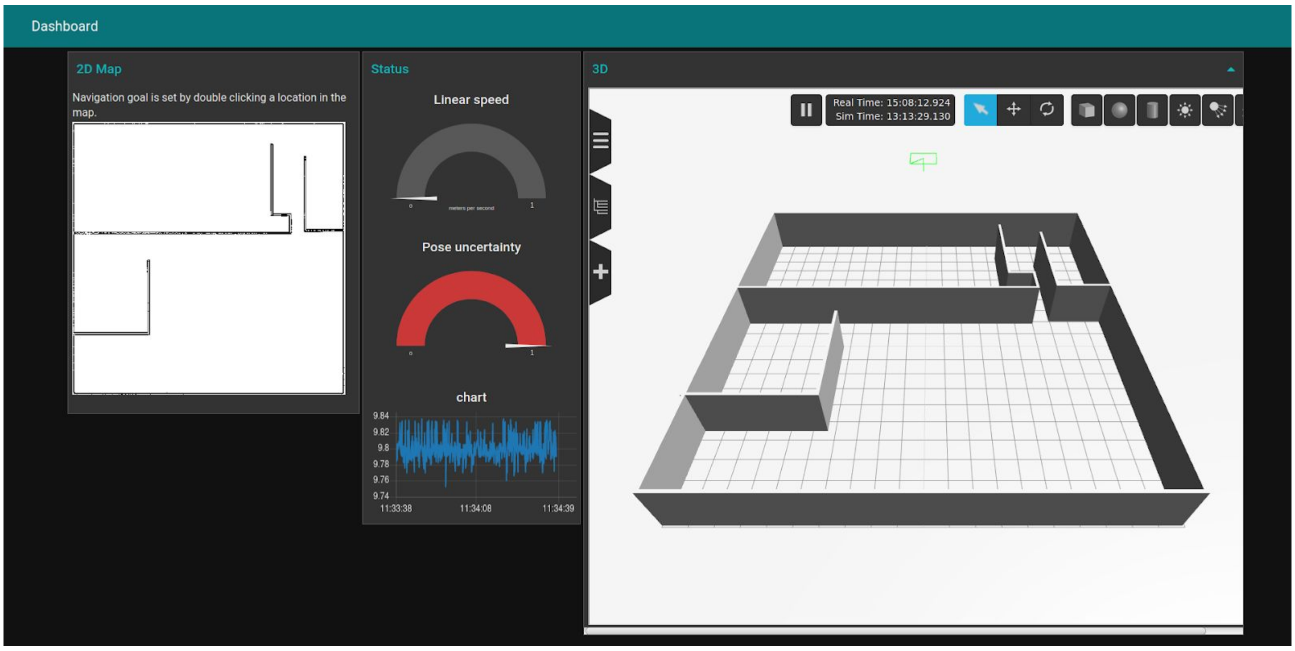
Aalto University, School of Electrical Engineering
Automation and Electrical Engineering (AEE) Master's Programme
ELEC-E8004 Project work course
Year 2020

Final Report

Project #1_3 Distributed Intelligent Production Involving Remote Actors



Date: 28.5.2020

Panu Salo
Niko Karhula
Juhani Lähde
Minh Duc Pham
Paavo Kajola

Information page

Students

Panu Salo

Niko Karhula

Juhani Lähde

Minh Duc Pham

Paavo Kajola

Project manager

Juhani Lähde

Official Instructor

Ronal Bejarano Rodriguez

Other advisors

-

Starting date

16.1.2020

Completion date

28.5.2020

Approval

The Instructor has accepted the final version of this document

Date: 28.5.2020

Abstract

The ELEC-E8004 Project Work is designed to give a team of Master's students a practical team-working environment while solving an automation challenge. The project was based in the Aalto Factory of the Future (AFoF) and in close cooperation with VX Lab in RMIT Australia. This project's purpose was originally to successfully utilize a MiR100 robot in a controlled manner while further implementing simulation possibility and secure a remote setup for interconnectivity and interoperability despite the geographical differences.

Normally the stationed manufacturing sites are being operated on-site with the compulsory presence of the operator. With the use of the open-source platform ROS (Robot Operating System), Gazebo simulator, OpenVPN and their supported distributed environment, this project introduces an innovative and flexible way to the manufacturing floor when it enables remote manipulation of AGVs without direct contact. Moreover, this solution requires little to none changes in the factory infrastructure and ready-to-be embedded into the available resources.

Due to covid-19, the physical MiR100 component in the lab was replaced by a simulation environment based on ROS, Gazebo, made by DFKI. The simulated MiR100 robot was localized and navigated in a static premade environment. The user and the robot are communicating locally through Gazebo and RViz interface or remotely through a web interface. The user provides the robot with missions while the robot returns the map data and the robot's operating parameters. Since ROS can only be installed on a few specific operating systems, the use of virtual machines with Debian/Linux operating systems installed offered clients with no access to such OS can still involve in the project development.

Communication protocol posed as one of the main challenges regarding this project. Because of the aim to operate without borders, the Internet as the medium of connection is the only viable option to secure the integrity and confidentiality of communication. Network and latency tests have been carried out and analyzed in order to assess the stability and the throughput of signal.

Table of Contents

1. Introduction

In the middle of the fourth industrial revolution, innovative solutions for agile production processes are in high demand as companies and factories worldwide are seeking new ways to improve not only working conditions but also productivity and production quality. In response to this need, this project offers a highly flexible production model which is customized to efficiently implement intelligent machinery (AGVs) with the help of digital systems. The element of real-time interactions are demonstrated throughout the project through the integration of industrial automation, information technologies and virtual manipulations.

AGVs are robots that are able to move by themselves under automatic control to execute the assigned tasks. The capabilities required for a self-propelled robot are the battery capacity that ensures the robot's function within the required time frame, the ability to detect, identify and avoid obstacles and the ability to compute complex algorithm programs in real time to respond promptly to the changes of the environment. Above all, efficient localization and navigation in order to recognize the direction of the robot is the most important. Orientation can be defined as a combination of three simultaneous activities, which are self-localization, path-planning and map building and interpretation respectively.

ROS is an open-source operating system, which is used for robot applications. Basically ROS has all the essential characteristics of an operating system such as the ability to perform multiple tasks in parallel, communicating and exchanging data between different tasks, publishing and subscribing functions, etc. ROS has been developed continuously by users for libraries and a wide extent of tools. Gazebo is an integrated part of ROS and serves as a powerful 3D simulator, thus capable of interacting with various robotic platforms. A combination of ROS and Gazebo has been widely used for their reliability, flexibility, robustness and the availability of the necessary features.

The findings of this project will be utilized as tools and frameworks for future experimentation and development on the physical components in physical environments. In addition to the completion of the course, this project serves as a stepping stone for further cooperation between Aalto Factory of the Future and VXLab in distributed automation. International collaborative projects are expected to bring fruitful results to real-life challenges.

2. Objective

The objective of this project was to study and manipulate self-propelled AGVs through their positioning and path-planning in a simulated industrial environment with remote access. The robot should be able to receive the remotely assigned missions, localize and navigate itself to the desired locations without interrupting the surrounding environment. Objectives were to formulate scenarios of control and setup of an AGV serving a distributed production system, and to design and implement an illustrative use case to demonstrate the scenarios.

With AFoF in Finland and VX Lab in Australia working in conjunction, the setup of a distributed intelligent production system which allows secure remote control was planned to be illustrated. A system prototype with in-depth technical data and instruction manual will be used to display different use cases for better understanding.

3. Project plan

The initial version of the project plan document set the main objectives as:

1. Setup of a secure IP-based communication channel between AFoF in Finland and RMIT VxLab in Australia;

2. Demonstrate remote control of MiR 100 AGV located in Australia from Finland.

Phases of the project were modulated as the planning phase (week 1-6), business aspect phase (week 6-11), development phase (week 7-14), testing phase (week 11-17) and final phase (week 17-21). Milestone M1 in the planning phase was creating a project plan, milestone M2 producing business assignments, milestone M3 achieving to move the robot at AFoF for the first time. In the original plan, milestone M4 was planned to be moving the Australian robot remotely for the first time, and milestone M5 was planned to be preparing technical data, video and instruction manual. It also included making a final gala presentation and final report.

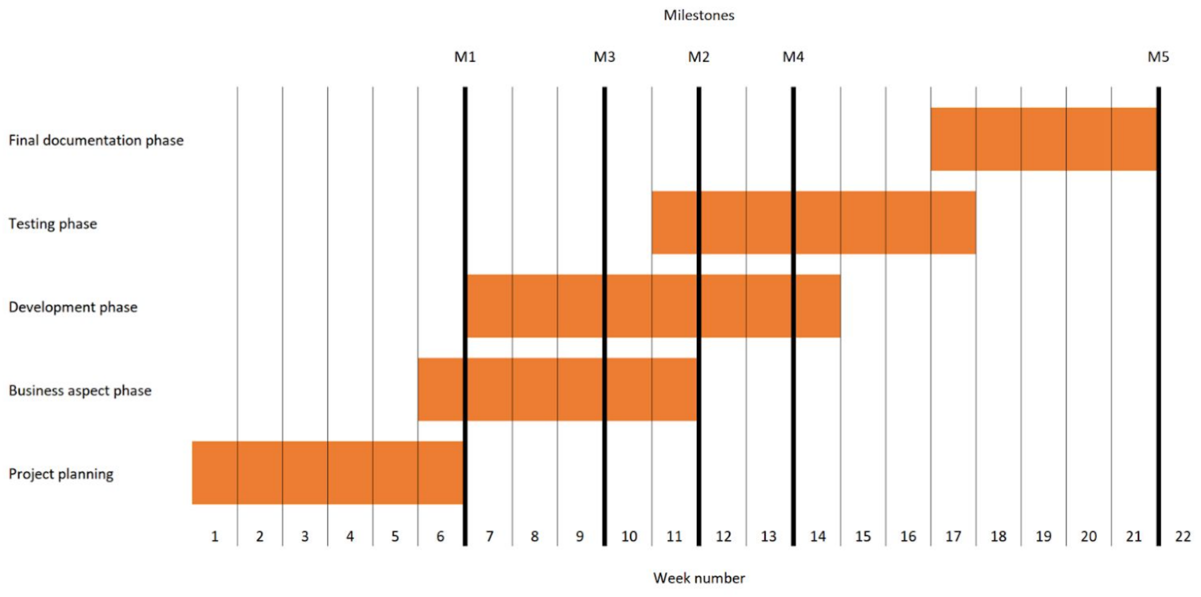


Figure 1. Original project plan's Gantt chart

Milestones M1, M2 and M3 were completed successfully before campus lockdown, but after week 12 there was no more access to the lab in Finland because of safety reasons. This meant redefining plans made in the project plan, and also cancelling some parts planned. Also in Australia the university was closed.

Parts of this project that were cancelled due to coronavirus included:

- further moving and control of MIR 100 AGV at AFoF in Otaniemi, Finland
- integration of MIR 100 with EnAS 61499 NxtStudio distributed automation system at AFoF in Otaniemi, Finland
- and remote moving and control of Australian MIR 100 AGV with networking: because campus closed also in Australia

It was decided in the group that the physical MiR100 component would be replaced by a simulation environment based on ROS, Gazebo, made by DFKI. All working was changed to remote working after week 12.

4. Early results

Once the robot had arrived, it had to be unboxed and commissioned. Commissioning the robot was an important exercise, which taught the team the basic operation of the robot. Initial tests regarding MiR 100 control and configuration were also performed before the campus closure. In addition, the team studied potential methods for information secure remote control and configuration of the robot. This chapter highlights some of the early results and knowledge gained during this phase.

4.1. REST API

The MiR 100 robot has an embedded CPU running a Linux-based operating system onboard. The operating system hosts a web server, which provides RESTful web service for control and configuration of the robot. This web service can be accessed through a wireless WiFi hotspot internal to the robot. Alternatively, an external WiFi hotspot may be used in a LAN to access the web service. One of the first milestones of the team was to control the robot by connecting to the internal hotspot from a laptop and triggering missions by sending requests to the REST API.

First, a map of the AFoF environment was created with tools offered by a web user interface hosted on the server of the onboard computer. Missions were then predefined within the web user interface. Simple Python script, acting as a REST API client and utilizing *requests* library, was written, and the predefined missions were executed by invoking RESTful web services from the robot by means of HTTP requests. Missions were successfully completed, and this marked the completion of one of the early milestones of the team.

Next, plans were made to extend the functionality of the simple Python script. First of all, the need to modify the script in such a way that REST API message structures could be stored in, for example, JSON format, and any REST API request could be sent without explicitly hard coding the processing of requests, was identified. Then, the next step would have been to extend the script with functionality to process end-user defined missions at higher levels of abstraction than offered by the web interface of the robot. Furthermore, the team had discussions about providing a graphical user interface for configuring and executing end-user defined missions.

Ultimately, the end-user would have accessed the RESTful web service remotely with a Python application, which would have invoked services in the remote MiR 100 by the REST API through a secure OpenVPN or IPSec tunnel. Unfortunately, the campus closure occurred before network tests or further progress with the Python script was made.

4.2. Communication protocol

A central challenge in the project was the setup of a connection between a remote site (MiR 100) and the end-user's client machine. Because the goal was to enable access to the remote site from very far away, the only viable option was to use the Internet as the medium of connection. The Internet is in general an unreliable and insecure medium for critical connections. However, low-level control of the robot and navigation are performed locally, and hence, the reliability requirements may be relaxed. On the other hand, the integrity and confidentiality of communication is paramount.

Mainly two options for secure connection were considered: OpenVPN and IPSec. A potential third option would have been WireGuard, which is a modern communication protocol for easy and secure VPN setup. WireGuard-enabled connection has possibly better performance than OpenVPN- or IPSec-enabled communication. However, a stable production release of WireGuard was released only after the team had begun the project. Table 1 illustrates the differences between OpenVPN, IPSec and WireGuard.

Protocol	**Overview**	**Security**	**Performance**
OpenVPN	Very secure and simple, but lacks some advanced features.	Considered very secure; no known vulnerabilities.	Great performance with UDP.
IPSec	Flexible, but also complex. Widely used in commercial solutions.	Likely very secure; no known vulnerabilities.	Excellent performance with IKEv2/IPSec; varying performance with L2TP/IPSec.
WireGuard	Novel solution, which aims to be simple, lean and performant.	Minimal attack surface, but not yet as thoroughly audited as the alternatives.	Excellent performance; faster reconnections compared to the alternatives.

Table 1. Comparison of different VPN protocols.

OpenVPN was chosen as the communication protocol because it has all the features that the team needed, and it is mature, very secure and relatively simple to set up.

4.3. Network tests

Latency and throughput capabilities of two Raspberry Pi´s using OpenVPN were measured. Script for testing was implemented with Python and provided by the instructor. The script executed measurement every 1 minute and wrote the results to a database file (SQL). The longest test executed in the Helsinki area was 360 minutes (6 hours) and the results are shown in Figure 2 and 3. Average latency was 12 ms and average throughput was 538 kB/s.

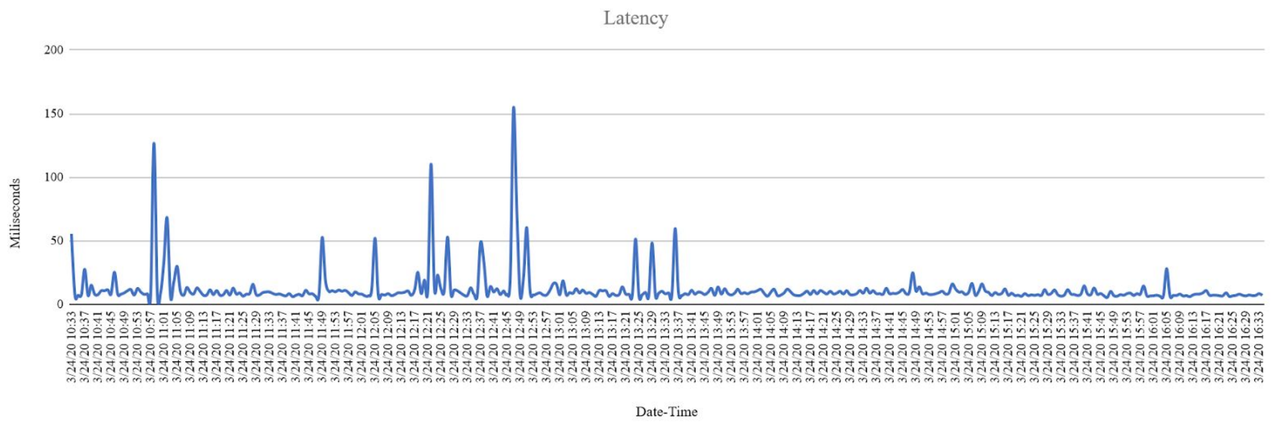


Figure 2. Measured latency using OpenVPN in the Helsinki area.

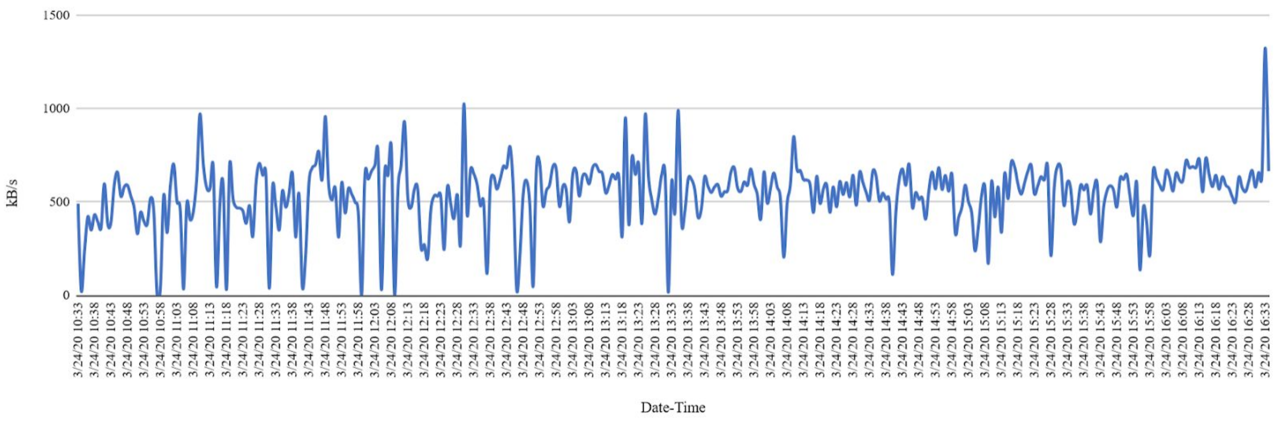


Figure 3. Measured throughput using OpenVPN in the Helsinki area.

Network capabilities between Finland (Helsinki) and Australia (Melbourne) were also measured using 1440 minutes (24 hours) test. The results are shown in Figure 4 and 5. Average latency was 358 ms and average throughput was 20 kB/s.

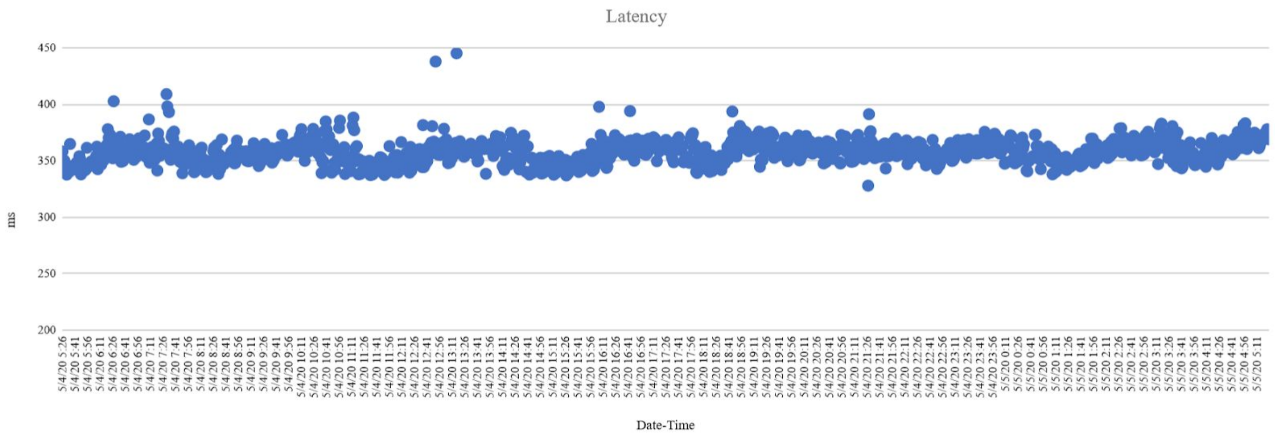


Figure 4. Measured latency using OpenVPN between Helsinki and Melbourne.

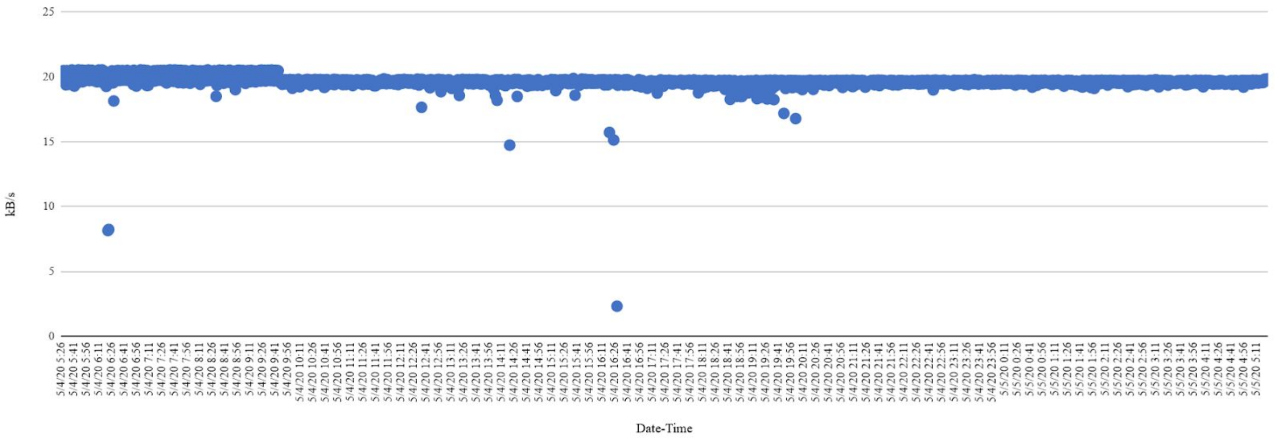


Figure 5. Measured throughput using OpenVPN between Helsinki and Melbourne.

As seen in Figures 2-5 there is a significant difference in the results when distance between measurement points differs. According to the results, network capabilities are drastically poor when geographical distance is enormous and this most probably prevents using our solution for sufficient remote control in large distances. More investigating to network solutions is needed, if remote controllability between Finland and Australia is desired.

5. Final system architecture

5.1. Description of the architecture

After the campus closure, the team had to revise the original plan for the system architecture. As mentioned earlier, access to AFoF was lost and any work with the physical setup (EnAS and MiR 100 robot) had to be cancelled. To accommodate this new situation, the team had two choices. Either setup ROS and use the ROS MiR 100 package maintained by DFKI to simulate the robot in a virtual Gazebo environment, and then setup a client machine for remote control and configuration of the simulated robot. Another option was to partially replicate the RESTful web service offered by the real robot and continue developing the Python REST API client script. This choice would have required a custom made graphical environment for visualization of the robot.

The team chose the ROS/Gazebo option because creating a graphical environment from scratch would have required skills the team did not possess and replicating the RESTful web service would have been a rather unfruitful task. Ideally, the RESTful web service would have been open source, and the team could have then utilized it while focusing on the remote control and configuration aspects.

The updated system architecture is depicted in Figure 6. Debian/Linux operating system was installed in a virtual machine, and the MiR 100 robot was then simulated in ROS/Gazebo. For remote control, the team could have set up a ROS installation on the client machine and created a custom ROS package for handling control and configuration of the simulated robot. Communication between ROS nodes in VPN should work sufficiently well, however, ROS can only be installed on a few specific operating systems. The architecture should be such that the virtual machine hosting the simulation is largely agnostic about the client machine. Hence, ROS should be separated from the remote control and configuration components in some way.

Fortunately, a solution for communicating with ROS nodes from outside ROS installation is provided by the ROS framework itself. Rosbridge suite consists of ROS nodes, which allow publishing and subscribing to topics, calling ROS services, and interacting with *actionlib* from outside ROS installation. These capabilities are realized by two libraries: *roslibpy* and *roslibjs*. The latter is a good choice for creating a web-based service for remote control and configuration of the robot. The advantage of a web-based service is that it requires minimal setup on the client machine; a web browser and OpenVPN client are the only requirements. Client side input handling is implemented using jquery.

On the server-side, a web server is needed for providing the web service. The team had limited experience with web technologies, which had to be considered when choosing appropriate web technologies for the provision of the service. Node-RED allows easy-to-learn flow-based web programming, and it is tailored for IoT and IIoT applications, which makes it a reasonable option for the project. Furthermore, Node-RED is built on Node.js[1], which is advantageous because the communication delay between Rosbridge and *roslibjs* stays low. In possible future developments this design choice should be useful in keeping separation between the client machine and the simulation, and thus, making it easier to avoid issues relating to communication delay.

[1] Node.js is a JavaScript runtime environment for executing JavaScript code on the server-side.

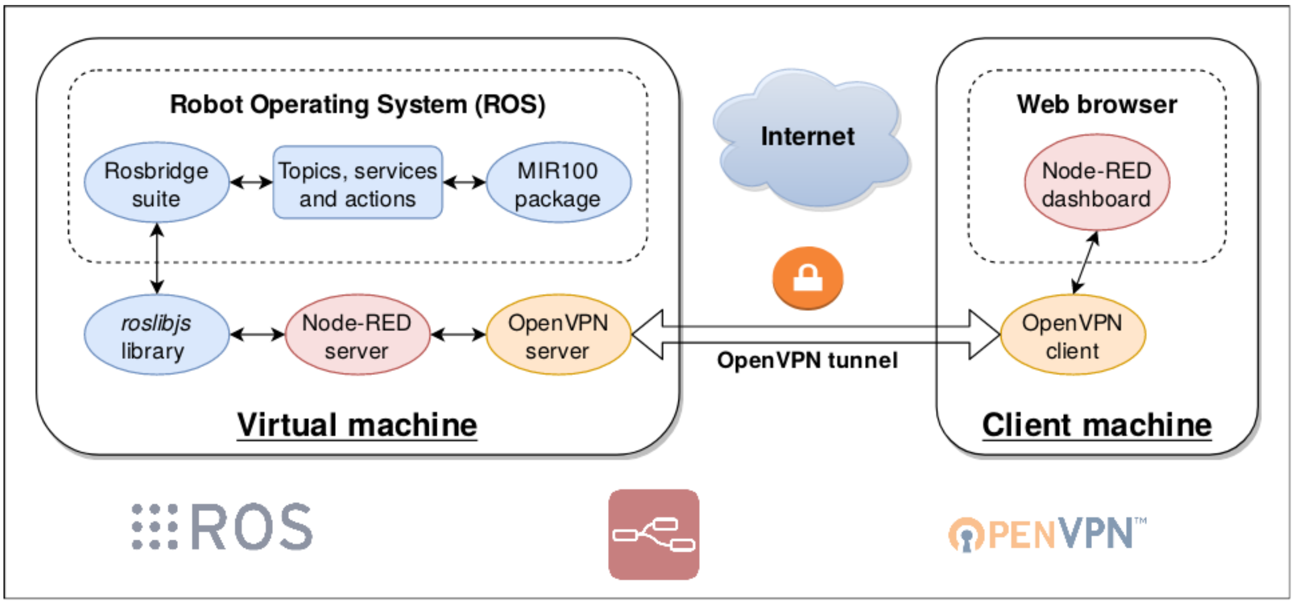


Figure 6. Final system architecture for remote control and configuration of a simulated MiR 100.

5.2. *Use cases: setup and control*

The robot is controlled by setting navigation goals on a 2d map. The client map is updated automatically each time the ros system sends the map update. The system supports dynamic map rescaling. Navigation goals imitate the rviz method of setting locations by clicking. The initial click (mousedown event) sets the navigation point. Dragging and then releasing the mouse click (mouseup event) sets the target direction the robot ought to face. A 3D view similar to the one in Gazebo on the host machine is available through GzWeb.

The robot's position is updated on the map every time a new position message is received. The robot simulation only publishes new pose messages when certain thresholds are exceeded, i.e. the robot must move a short distance before an update is sent. Robot motion is not interpolated client side, which results in a slightly choppy visual representation of the robot's current position.

Navigation messages are sent to the simulated system immediately when the mouse click is released. The network is insignificant compared to the robot's slow navigation planning time. The network latency would cause problems with direct control with e.g. joystick, as increasing distance would cause higher latencies. When real time response is expected, even relatively minor latencies may cause user dissatisfaction.

The instability of the simulation prevents extensive testing of the control system. The simulated robot is greatly hindered by even simple obstacles, and especially the simulated virtual walls, which confuse the navigation system. As the robot struggles with surrounding obstacles, the covariance of its pose estimate increases. As the robot becomes increasingly unsure of where it is, the visual representation starts warping.

System setup is relatively simple. Several nodes need to be run on the server machine. These nodes include robot simulator, navigation stack and rosbridge. Configuration for these nodes is included in the launch files. If roscore is not already running when a node is launched, it is automatically launched as well. Subsequent node launches do not affect roscore. Multiple instances of roscore are not launched.

Several packages need to be installed on the server before use. Node.js is installed first, followed by Node-RED. An additional ROS integration package is installed next. For 3D viewing, GzWeb

(Gazebo web client) must be installed. The client does not need to manually install any packages, and only requires a web browser.

The robot simulation nodes are launched in the following order: The Gazebo environment is launched first. When it is ready, the simulation is unpaused via either the Gazebo GUI, or the terminal. This step is followed by robot localization. The MIR navigation package is launched with arguments for the robot's initial pose. The final step is to launch the navigation planner. Optionally, server-size visualization and navigation control may be attained by launching rviz. Rviz provides a 2D representation of the environment and implements the capability of setting navigation points in the same manner as our implementation. Rviz requires installation and configuration, whereas our solution is (from the client's perspective) plug and play.

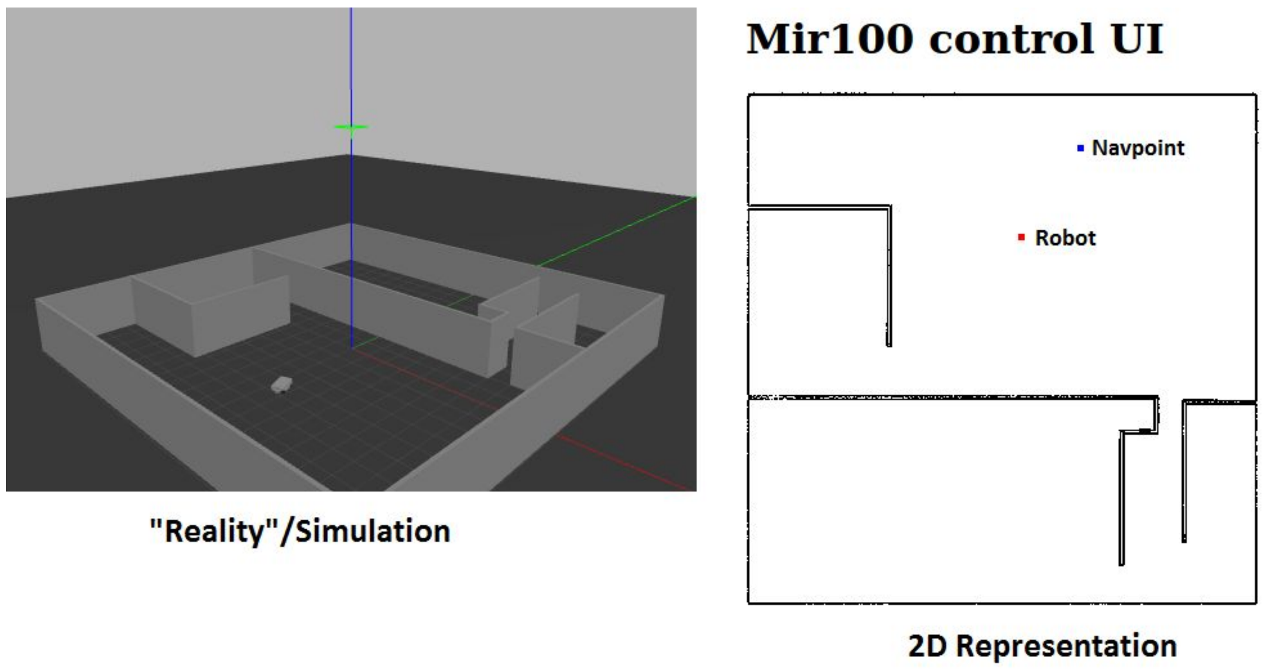


Figure 7. UI (user interface) in the simulation

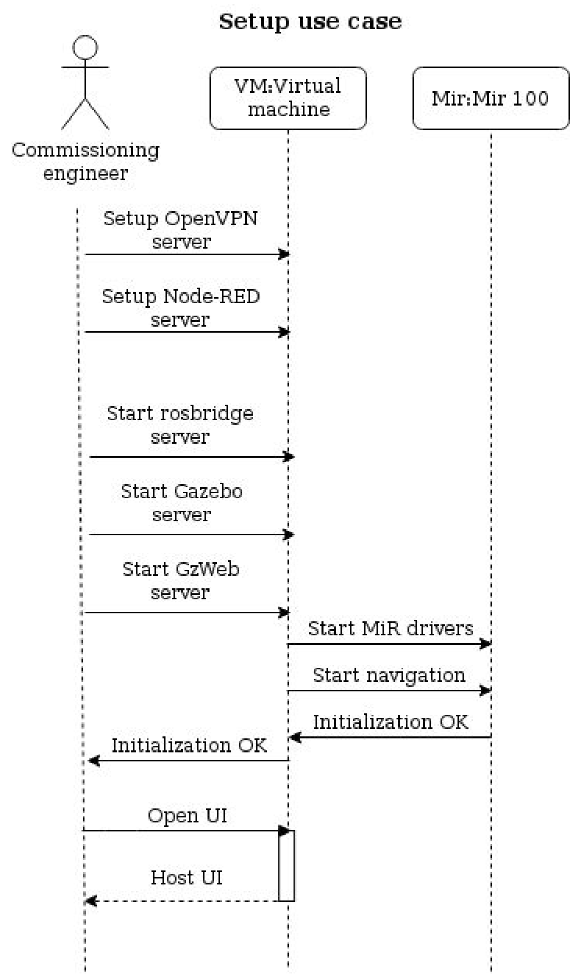


Figure 8. Setup use case diagram (commissioning engineer's view)

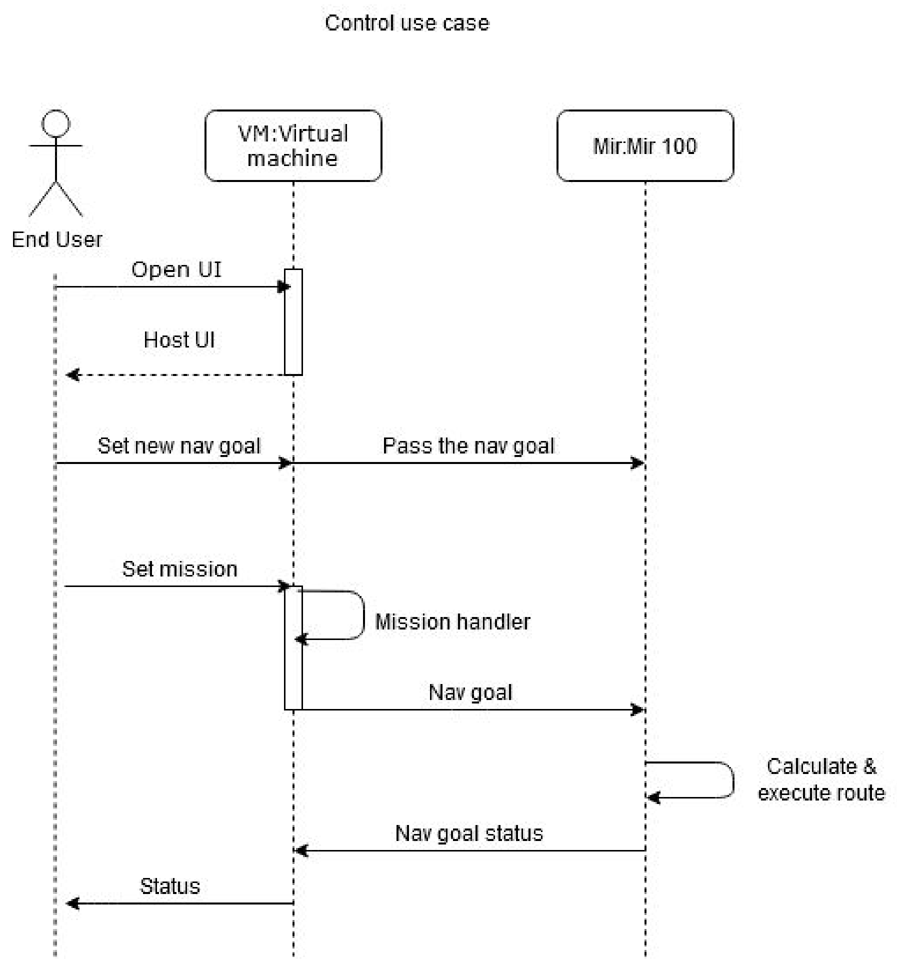


Figure 9. Control use case diagram (end user's view)

6. Reflection of the Project

The group's working proceeded well according to the project plan until week 12, when covid-19 forced Aalto University campus. Access to the lab's MiR100 robot was not possible after this. Instead, remote working with a simulation was started. Physical meetings were then changed to Microsoft Teams meetings. Online repositories were used to store text and code of the group for it to work with. Covid-19 was not considered as a risk in the risk table of the project plan, so it surprised the group.

6.1. Achieving objectives

Before covid-19 pandemic, the group achieved well its objectives (milestones M1-M3, project plan, business assignments, moving MiR100 in the lab), and it made promising progress. Covid-19 caused campus closure after week 12 in Aalto University, which caused the group to redefine its goals.

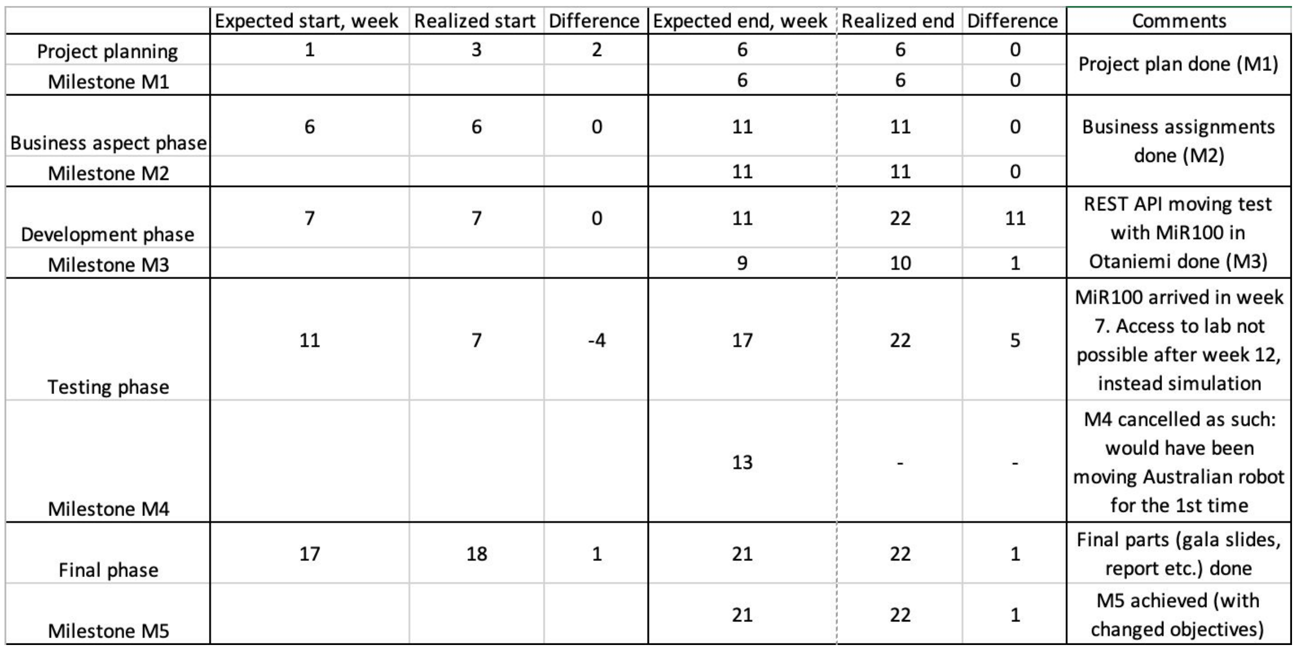
As the ultimate objective, there was replacing the MiR100 component by a simulation environment based on ROS, Gazebo, made by DFKI. Other possibilities were also discussed in the group, but since there was no other open-source (ROS) simulation environment of MiR100 available, or it could not be found, it was chosen as the replacement component of the physical 'real' MiR100 AGV.

Therefore, the global covid-19 pandemic caused changes to the project. One of things cancelled in the project was obviously further moving and control of the physical MiR 100 AGV at AFoF in Otaniemi, Finland, because Aalto University campus was closed for safety reasons. The plans to integrate MiR 100 with EnAS 61499 NxtStudio distributed automation system at AFoF, were cancelled for the same reasons. Because campus was also closed in Australia, remote moving and control of Australian MiR 100 AGV with networking (OpenVPN or IPSec) was not possible. OpenVPN was chosen for the choice to get remote connection for the ROS simulation.

The group achieved well its objectives regarding the controlling of the ROS Gazebo MiR100 simulation of DFKI (see previous chapters for more information).

6.2. *Timetable*

The timeline and schedules before week 12's campus closure were met well. MiR100 did not arrive until early February, so all work before the robot arrival was planning and independent learning. Business phase, which took place between late February and early March, consumed a bit more time than expected, but it ended as planned. In the end, the changed objectives in the group to control ROS Gazebo simulation were met well. Development and testing phases continued longer than anticipated, but however, it is not easy to separate the testing phase, development phase and final phase completely from each other.

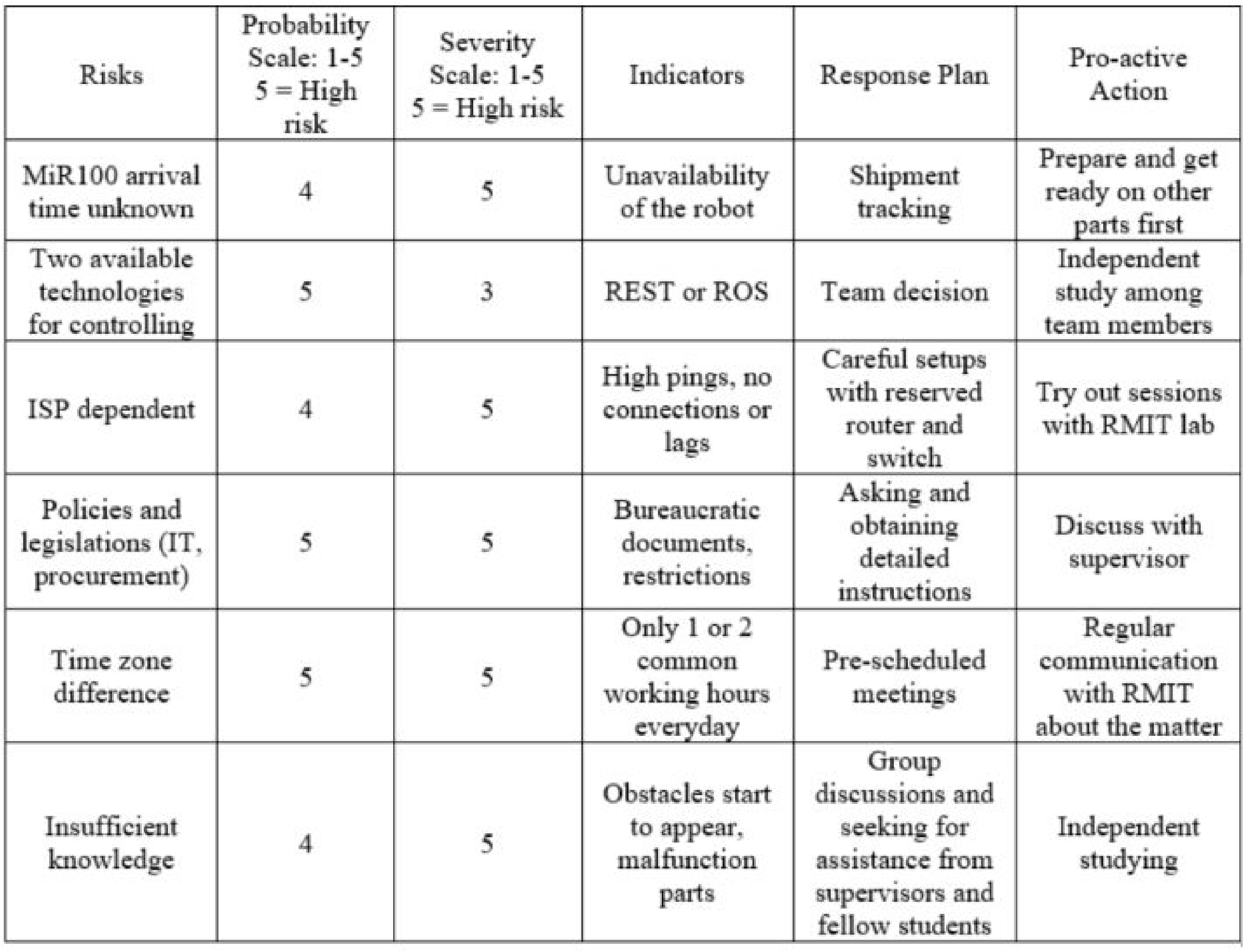


	Expected start, week	Realized start	Difference	Expected end, week	Realized end	Difference	Comments
Project planning	1	3	2	6	6	0	Project plan done (M1)
Milestone M1				6	6	0	
Business aspect phase	6	6	0	11	11	0	Business assignments done (M2)
Milestone M2				11	11	0	
Development phase	7	7	0	11	22	11	REST API moving test with MiR100 in Otaniemi done (M3)
Milestone M3				9	10	1	
Testing phase	11	7	-4	17	22	5	MiR100 arrived in week 7. Access to lab not possible after week 12, instead simulation
Milestone M4				13	-	-	M4 cancelled as such: would have been moving Australian robot for the 1st time
Final phase	17	18	1	21	22	1	Final parts (gala slides, report etc.) done
Milestone M5				21	22	1	M5 achieved (with changed objectives)

Table 2. Timetable of the project

6.3. *Risk analysis*

The following graph shows the anticipated risks from the planning phase at the beginning of the project.



Risks	Probability Scale: 1-5 5 = High risk	Severity Scale: 1-5 5 = High risk	Indicators	Response Plan	Pro-active Action
MiR100 arrival time unknown	4	5	Unavailability of the robot	Shipment tracking	Prepare and get ready on other parts first
Two available technologies for controlling	5	3	REST or ROS	Team decision	Independent study among team members
ISP dependent	4	5	High pings, no connections or lags	Careful setups with reserved router and switch	Try out sessions with RMIT lab
Policies and legislations (IT, procurement)	5	5	Bureaucratic documents, restrictions	Asking and obtaining detailed instructions	Discuss with supervisor
Time zone difference	5	5	Only 1 or 2 common working hours everyday	Pre-scheduled meetings	Regular communication with RMIT about the matter
Insufficient knowledge	4	5	Obstacles start to appear, malfunction parts	Group discussions and seeking for assistance from supervisors and fellow students	Independent studying

Table 3. Anticipated risks from planning phase

All of the above risks are somewhat materialized except for the uncertainty in arrival time of the MiR100 robot since it came on time as agreed with the vendor. The team had to change the technical methodology in the project from REST to ROS because of the change of objectives. This is a time-consuming process because of the limited time-frame and the unfamiliarity of the targeting method, which displays another forecasted risk, the insufficient knowledge. The ISP dependent appears as unfavorable latency results came back when testing the connection between Finland and Australia, however it is note-worthy that the results within the Helsinki region were rather promising.

Policies and legislations were reflected through the difficulties in setting up the private network hotspot in the AFoF premise. Even when everything moved online, this risk was still troublesome when the virtual machine on Microsoft Azure with Aalto license had to change its access protocol every now and then. The time zone difference was one of the first risks to emerge from the project. Early meeting in the morning here when everyone was still passive happened to be an after-work meeting over in Australia, when everyone was tired after a long day.

An unexpected risk for this project was covid-19, of course. No one in the group expected that the whole project would change for remote working and that people would not have access to the lab. Also, the closure of Australian university, RMIT's VxLab, was also an unexpected event. In order to combat the change in team-working conditions, changes of project objectives and project plan were to be introduced.

In general, all the risks were minimized and resolved through effort, careful discussions, self-study and suggestions from the supervisor.

6.4. Project Meetings

Physical project meetings at AFoF were arranged regularly since the beginning. They were arranged once in a week, almost every week. In the beginning, agendas and meeting memos were recorded regularly. During the business phase between late February and early March, memos were less recorded, since most of the focus then was for the business pitching etc. assignments. Telegram group was formed already in the beginning and served well as a discussion forum within the group. Email (Aalto webmail) communication was also used for formal remarks and meeting bookings.

After week 12's campus closure in Otaniemi, meetings were started to be arranged as Microsoft Teams meetings. Meetings were arranged weekly at Teams. Agendas and memos were being focused more in order to maintain team integrity since no face-to-face interactions could be put in place. On top of that, individual tasks related to the project were agreed and assigned at the end of every virtual meeting. These were written to excel sheets for easy modifications and follow-up.

Project meetings appeared to be an irreplaceable part of any project. The whole team would be lost without the regular check-ups, the iterative process of tasks assignment, the discussions towards dead-ends, the suggestions provided by the supervisors and especially the short term goal which materialized all the ideas and thinking the team had.

6.5 Repositories

Google Drive and Aalto University's Gitlab (version.aalto.fi/gitlab) were used as project work's repositories (text, code, files, etc), and this worked well. Memos and meeting agendas, and also presentation slides, reports and document drafts were saved for Google drive folders for people in the group to work with. Gitlab is dedicated solely for the purpose of storing code, project structures and implementation guidelines. Furthermore, MIT license was used to license our repository on GitHub. Future development can utilize these repositories as the foundation for research.

The use of Trello, which is an online dashboard platform for status updating, progress checking, record keeping and future planning with the fellows from RMIT, was also introduced. However because of the pandemic, the communication between teams are rather limited and we communicated mainly through Telegram and Teams calls. The group learnt how to organize project organised meetings with an agenda.

7. Discussion and Conclusions

During the project, the team learned the importance of frequent meetings to keep track of progress and orchestrate future tasks and milestones. Intermediate results and knowledge was shared during the meetings, which helped the group members to keep a holistic understanding of the progress. Without frequent meetings, it is easy to become too focused on a narrow topic and lose sight of the big picture.

The ultimate goal of this project is to integrate flexible concepts in industrial automation. Even though the execution and content of the project have changed quite dramatically, however with this pandemic situation, the importance of this project is further emphasized. In the digital age we are living in today, where uncertainty and drastic change are to be expected, AGVs can help answer the call for increased interoperability and flexibility.

This project also brought out the benefits of modeling and simulation in engineering. With lots of obstacles such as limited access to resources, geographical differences and global crisis, simulation plays a pivotal role in developing, validating and predicting the feasibility of projects. This project's success is a solid indication of the undeniable potential of simulation in research and development.

In conclusion, this project has been a fruitful journey. Not only did it come up to the expectation, it has successfully served its purpose as a foundation for further research in the Aalto Factory of the Future. Teamwork, disciplines, problems-solving and brainstorming just to name a few have helped every team member to gain memorable and valuable experience.

References

http://wiki.ros.org/mir_robot

https://github.com/dfki-ric/mir_robot

https://requests.readthedocs.io/en/master/

https://restoreprivacy.com/openvpn-ipsec-wireguard-l2tp-ikev2-protocols/

https://wiki.ros.org/rosbridge_suite